ORSAY
N° d'ordre : 568

THESE

PRESENTEE

A L'UNIVERSITE DE PARIS - SUD
CENTRE D'ORSAY

POUR OBTENIR

LE TITRE DE DOCTEUR INGENIEUR
Spécialité : ELECTRONIQUE
Sous-spécialité : TRAITEMENT DE L'INFORMATION

par

Didier JUVIN
Ingénieur I.S.E.P.

CONTRIBUTION A LA RECONNAISSANCE AUTOMATIQUE DES IMAGES APPLIQUEE A LA ROBOTIQUE

Soutenue le 31 janvier 1983 devant la Commission d'Examen

MM.	P. DEBRAINE	Président
	B. DUPEYRAT	
	O. FAUGERAS	Examinateurs
	G. STAMON	
	J.P. WATTEAU	

UNIV. LIBRARY

ORSAY
N° d'ordre : 568

THESE

PRESENTEE

A L'UNIVERSITE DE PARIS - SUD
CENTRE D'ORSAY

POUR OBTENIR

LE TITRE DE DOCTEUR INGENIEUR
Spécialité : ELECTRONIQUE
Sous-spécialité : TRAITEMENT DE L'INFORMATION

par

Didier JUVIN
Ingénieur I.S.E.P.

CONTRIBUTION A LA RECONNAISSANCE AUTOMATIQUE DES IMAGES APPLIQUEE A LA ROBOTIQUE

Soutenue le 31 janvier 1983 devant la Commission d'Examen

MM.	P. DEBRAINE	Président
	B. DUPEYRAT	
	O. FAUGERAS	Examinateurs
	G. STAMON	
	J.P. WATTEAU	

DIDIER JUVIN

CONTRIBUTION TO AUTOMATIC IMAGE RECOGNITION APPLIED TO ROBOT TECHNOLOGY

SUMMARY : This paper describes a method for the analysis and interpretation of the images of objects located in a plain scene which is the environment of a robot.

The first part covers the recovery of the contour of objects present in the image, and discusses a novel contour-following technique based on the line arborescence concept in combination with a "cost function" giving a quantitative assessment of contour quality. We present heuristics for moderate-cost, minimum-time arborescence coverage, which is equivalent to following probable contour lines in the image. A contour segmentation technique, invariant in the translational and rotational modes, is presented next.

The second part describes a recognition method based on the above invariant encoding: the algorithm performs a preliminary screening based on coarse data derived from segmentation, followed by a comparison of forms with probable identity through application of a distance specified in terms of the invariant encoding.

The last part covers the outcome of the above investigations, which have found an industiral application in the vision system of a range of robots. The system is set up in a 16-bit microprocessor and operates in real time.

REMERCIEMENTS

Que Monsieur Pierre DEBRAINE, qui a assuré la direction de ce travail et qui n'a cessé de me manifester sa bienveillante attention tout au long de son élaboration, veuille bien trouver ici l'expression de ma profonde reconnaissance.

Je tiens à remercier Monsieur le Professeur Georges STAMON, Professeur à l'Université de Besançon, pour l'intérêt qu'il a manifesté pour mon travail. Que sa gentillesse soit ici honorée.

Je tiens à remercier vivement Monsieur Jean-Pierre WATTEAU, Directeur de l'ENSEA, ainsi que Monsieur Olivier FAUGERAS, Ingénieur de recherches à l'INRIA qui m'ont fait l'honneur de participer au jury de cette thèse.

Cette étude a été effectuée au Département d'Electronique et d'Instrumentation Nucléaire grâce à l'accueil bienveillant de Monsieur J. WEILL, Chef du Département et de Monsieur Y. AMRAM, Chef du Service d'Instrumentation pour la Recherche au D.E.I.N.

Monsieur B. de COSNAC, Adjoint au Chef du S.I.R. m'a prodigué durant ces travaux conseils et encouragements. Je l'en remercie sincèrement.

Je dois une reconnaissance particulière à Monsieur B. DUPEYRAT pour l'encadrement technique amical et efficace qu'il a bien voulu m'apporter, ainsi que pour avoir bien voulu faire partie de ce jury en étant le rapporteur de commission.

Je tiens à remercier tous les membres du DEIN et tout particulièrement N. NARDON, L. ANDRIAMAMPIANINA, E. CSAKVARY, M. MARCEAU, A. BASIRE et G. MANSUY pour leur aide sympathique dans une ambiance chaleureuse, ainsi que tous ceux et celles qui ont collaboré à ce travail pour sa réalisation et publication.

Je n'oublie pas enfin combien ont compté pour moi la sollicitude et la compréhension dont ont fait preuve mes parents et amis.

SOMMAIRE

- RESUME

- CHAPITRE I - INTRODUCTION

I.1. Généralités : Intelligence artificielle et robotique
I.2. Notre étude : . Principe et objectifs
. Vers un système de vision de haut niveau
. Position du problème
I.3. Plan du mémoire

PARTIE A : TRAITEMENT DES IMAGES

- CHAPITRE II - LA VISION ET L'ORDINATEUR

II.1. Introduction
II.2. Perception en deux dimensions
. Capteurs, digitalisation
II.3. Perception en trois dimensions
. Pseudo 3D
. Capteurs, digitalisation
II.4. Descripteurs : extraction de contour
II.5. Les algorithmes traditionnels d'extraction de contour
. Seuillage, transformée de Fourrier, convolution, gradient, ...
II.6. Le codage du contour
. par coordonnées, par marqueurs, FREEMANN
. compression

- CHAPITRE III - NOTRE METHODE : SUIVI DYNAMIQUE DU CONTOUR

III.1. Principe : Etude du gradient dynamique
III.2. Méthode proposée
. Arborescence
. Principe du coût associatif
. Détermination de la fonction de coût
. Heuristiques

III.3. Application de la méthode au suivi de lignes de contour dans une image : Présentation de l'algorithme

III.4. Cas singuliers ou ambigus
. Aiguillages
. Reflets, ombres

III.5. Résultats

PARTIE B : RECONNAISSANCE DES IMAGES

- CHAPITRE IV - SEGMENTATION - COMPRESSION

IV.1. Introduction

IV.2. Segmentation linéaire du contour
. Méthode d'extrapolation par recouvrement

IV.3. Codage invariant des segments
. Le codage polaire relatif

IV.4. Compression
. Définition des gros segments

- CHAPITRE V - RECONNAISSANCE

V.1. Deux méthodes traditionnelles de reconnaissance
. Moments invariants, masque

V.2. Méthode proposée :
. Sélection par coïncidence de gros segments
. Corrélation sur la représentation invariante
. Méthode retenue : combinaison heuristique des deux méthodes

V.3. Résultats

V.4. Conclusion

PARTIE C : UNE APPLICATION INDUSTRIELLE

- CHAPITRE VI - LE SYSTEME ANIMA

  VI.1. Présentation
  VI.2. Description matérielle
  VI.3. Structure logicielle
  VI.4. Utilisation
  VI.5. Un exemple d'application : saisie automatique d'objets sur un tapis roulant

- CONCLUSIONS

- BIBLIOGRAPHIE

- ANNEXES

  . Annexe A - Un système de mesure 3D : triangulation par laser
  . Annexe B - Application de l'algorithme de suivi de contour à une image 3D
  . Annexe C - Distance Euclidienne sur le codage développé
  . Annexe D - Extraits du programme

INTRODUCTION

CHAPITRE I : INTRODUCTION

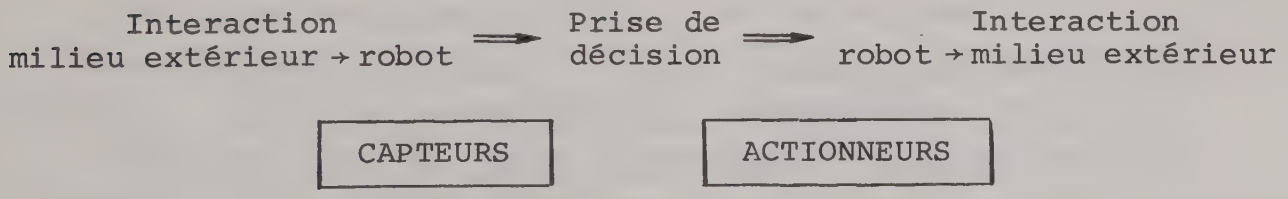
I.1. GENERALITES : INTELLIGENCE ARTIFICIELLE ET ROBOTIQUE

Au cours de la constante mutation caractérisant la société humaine, le phénomène de robotisation prend une ampleur que n'a connu aucun des autres vecteurs d'automatisation.

Depuis déjà une dizaine d'années, le robot-automate-programmable peut se substituer à l'homme dans certaines fonctions musculaires pour exécuter des travaux à caractère répétitif.

L'évolution conjuguée des techniques tant en informatique, en électronique qu'en mécanique permet de créer des systèmes de plus en plus élaborés capables d'appréhender l'environnement, de prendre des décisions, et d'agir sur cet environnement de manière appropriée pour réaliser une action planifiée uniquement dans sa finalité, c'est-à-dire sans que l'opérateur n'aie à prendre égard des difficultés pouvant se présenter lors de l'exécution, qui devront être analysées et résolues de manière automatique.

Cette adaptativité peut être évaluée en un critère d'"intelligence" de cet outil universel qu'est le robot. La robotique consiste donc en l'association d'un actionneur ou manipulateur (appelé plus communément et par abus de langage "robot") avec un système dit intelligent interprétant les données issues du milieu extérieur.



L'homme devant, bien-sûr rester maître de ce schéma et pouvoir, à tout moment, intervenir sur le robot.

Nous allons hiérarchiser arbitrairement la notion de capteur ou d'actionneur en fonction de la difficulté de traitement nécessaire à :

- l'interprétation des informations (capteur)
- la commande de manipulation (actionneur)

I.1.1. Interactions milieu extérieur → robot

* Capteurs simples

• Interrupteurs ou palpeurs délivrant des informations en tout ou rien : interactions issues du contact avec un objet ou rencontre d'une butée géométrique.

• Codeurs indiquant la localisation physique du robot: information de retour de position (capteurs potentiométriques, codeurs optiques).

• Capteurs exprimant l'interaction liée à la vitesse de déplacement de l'actionneur : information de retour de vitesse (variation temporelle de la position).

• Capteurs exprimant l'interaction liée aux effets de la pesanteur et de l'inertie mécanique sur l'actionneur (jauges de contraintes ou interprétation des couples moteur).

* Capteurs plus évolués

• Capteurs d'approche indiquant la proximité d'un obstacle sans pouvoir l'identifier (proximètre à ultra-sons, à diode laser).

• Capteurs de vision locale rapides permettant d'effectuer une analyse visuelle élémentaire dans le but de corriger immédiatement le mouvement (par exemple : mesure grossière de distance entre la pince du robot et l'objet à saisir). Ces capteurs sont des caméras à faible résolution (typiquement 64 x 64).

* Capteurs de haut niveau

Il s'agit de capteurs faisant réellement intervenir des notions d'intelligence pour interpréter les données fournies par ceux-ci.

• Utilisation du toucher (peau artificielle) dans la mesure où une identification de la forme palpée est possible.

• Capteurs de vision globale : analyse de scènes en 2D ou 3D. A partir des informations données par un capteur (généralement, une ou plusieurs caméras), on est capable de reconnaître et de localiser une forme par référence à une modélisation interne.

• Reconnaissance de la parole et interprétation du langage : la machine est capable de reconnaître et d'analyser des ordres énoncés oralement en langage naturel.

I.1.2. Interactions robot → milieu extérieur

De même, pouvons-nous classer les commandes en fonction de la complexité de traitement qu'elles nécessitent.

* Commandes simples

• Commandes liées au positionnement statique du robot : asservissements des moteurs, modélisation géométrique du robot et transformation de coordonnées (transcription de la localisation désirée de l'effecteur en valeurs de consigne à appliquer à chaque articulation du robot).

* Commandes plus évoluées

• Commandes liées à l'aspect cinématique ou dynamique du robot (c'est-à-dire faisant intervenir les notions de vitesse ou d'accélération dans la commande du mouvement).

* Commandes de haut niveau

• Cybernétique du mouvement : il s'agit de commandes évoluées capables de faire des prévisions, de prendre des décisions en fonction du contexte. Ces commandes font réellement intervenir la notion d'intelligence artificielle. (Elaboration de stratégie, détermination du plan à mettre en oeuvre pour aboutir à la réalisation de l'action désirée, création de la trajectoire optimale évitant les obstacles).

I.1.3. Observations

Notre classification pourra être qualifiée d'arbitraire. Néanmoins, celle-ci fait apparaître clairement l'évolution de la recherche : on a commencé par relier des capteurs simples à une commande simple (c'est le cas, par exemple, des robots-automates-programmables). L'évolution des techniques a permis de progresser tant dans le domaine des capteurs que dans celui de la commande. Cette dualité est indispensable : on ne peut imaginer de robot à commandes évoluées dépourvu de capteurs ayant un degré d'évolution équivalent, et réciproquement, sous peine d'une dégradation sensible des performances potentielles du robot.

Cette dernière remarque nous a incitésà étudier un senseur de haut niveau particulièrement utile à l'homme contemporain : la vision.

I.2. NOTRE ETUDE : PRINCIPES ET OBJECTIFS

I.2.1. Vers un système de vision de haut niveau

Notre travail a porté sur l'analyse et l'interprétation d'images en vue d'applicationsà la robotique. Afin que cette étude ne reste pas purement théorique ni académique, nous avons décidé de restreindre le problème d'analyse de scènes dans un environnement industriel et de tendre à répondre ainsi à un besoin réel.

Dans les conditions industrielles, l'obtention d'une bonne image pose des problèmes parfois insurmontables : on n'est pas toujours maître de l'éclairage ; des parasites électriques viennent perturber la perception ; les objets sont mal distingués de leurs supports (tapis roulant ou autre).

C'est pourquoi, il nous a paru primordial de concevoir un système capable de s'adapter à une telle ambiance afin de faciliter l'insertion de celui-ci dans la chaîne de production. Nous avons ainsi conçu et réalisé un système de vision "adaptatif" capable, dans un environnement peu favorable, de reconnaître une dizaine d'objets d'après leurs formes, de localiser ces objets et d'indiquer les points de prise en un temps compatible avec la plupart des opérations effectuées en robotique (saisie, assemblage), c'est-à-dire de l'ordre de deux à trois secondes. L'aspect adaptatif de ce système a été la contrainte majeure imposée à l'étude.

I.2.2. Position du problème

Dès que l'on désire réaliser un système fonctionnant "en ligne", une des premières questions que l'on est amené à se poser est quelle est la proportion logiciel-matériel à adopter ?

En règle générale, on admet ce tableau :

	Solution matérielle	Solution logicielle
1	. Temps d'exécution rapide	. Temps d'exécution lent
2	. Long à mettre en oeuvre	. Rapidement mis en oeuvre
3	. Rigidité (difficile à modifier)	. Souplesse - flexibilité
4	. Fort prix de revient	. Faible prix de revient*
5	. Dépannage difficile	. Dépannage facile (si programme bien structuré)

* Si on dispose déjà d'une structure informatique d'accueil.

Les contraintes 3 et 4 nous ont orientés vers le choix d'une solution essentiellement logicielle. Il peut sembler étrange d'exclure, à priori, les solutions cablées pour un système devant traiter une quantité d'informations aussi conséquente à cause du temps de traitement plus important, (une image de télévision nécessite pour son codage de 32 K octets à 600 K octets - cf. II.2.).

Cependant, l'utilisation de microprocesseurs ayant des performances accrues, tels le 8086 de INTEL auquel va s'ajouter une nouvelle génération (80186 et 80286) a permis, en concordance avec des algorithmes heuristiques performants, de rendre une telle solution réalisable en logiciel.

I.3. PLAN DU MEMOIRE

Nous développerons notre exposé en trois grandes parties :

- Dans un premier temps, nous nous attacherons à l'étude des images et à un descripteur particulièrement intéressant : le contour.

Après avoir exposé certaines méthodes traditionnelles d'extraction de contour, nous présenterons notre algorithme de suivi dynamique.

- Une deuxième partie est consacrée à la reconnaissance des images d'après leur contour : nous développerons sur une technique de compression d'information puis consacrerons un chapitre à la reconnaissance.

- La dernière partie décrit le système de vision tel qu'il sera proposé, ainsi qu'un exemple d'application à la reconnaissance d'objets sur un tapis roulant.

Enfin, nous tirerons nos conclusions sur l'ensemble de ce travail.

PARTIE A

TRAITEMENT DES IMAGES

CHAPITRE II : LA VISION ET L'ORDINATEUR

II.1. INTRODUCTION

Tout être vivant est doté d'un grand nombre de senseurs (oeil, oreille, antenne, odorat, peau,...) permettant la perception du milieu environnant. Il est, en effet, vital qu'il puisse instantanément acquérir et répertorier toute sensation afin d'agir en conséquence après avoir classé la forme perçue comme :

- nécessaire,
- désirable,
- indifférente,
- gênante,
- dangereuse,
- mortelle.

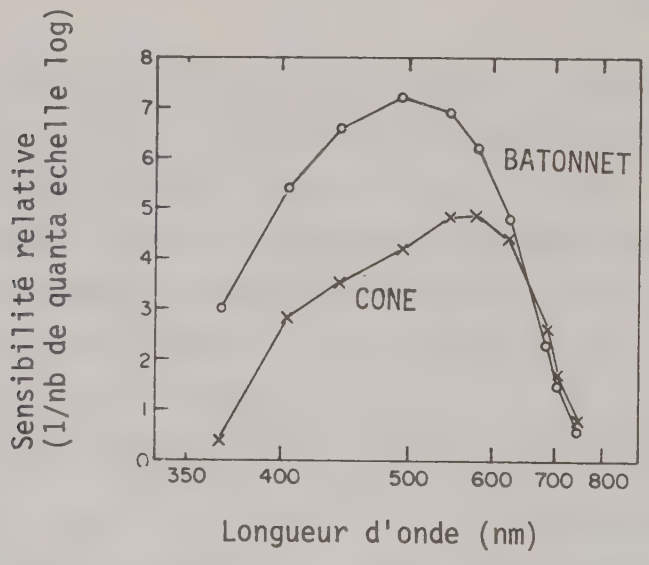
Le spectre de la lumière visible (c'est-à-dire l'énergie que l'homme et beaucoup d'animaux peuvent percevoir avec les yeux) est, de loin, la plus riche source d'informations sur l'environnement. Le système visuel humain est composé d'environ 100 millions de capteurs élémentaires (cônes et bâtonnets) dans chaque oeil, plusieurs millions de neurones transmettent les informations perçues par ces capteurs au cortex. On estime, grossièrement, que la moitié du cerveau (12×10^{12} neurones) est concernée par la perception et principalement par la vision. Les ordinateurs les plus évolués apparaîssent bien petits devant ces possibilités !..

Nous sommes tous capables de percevoir et de reconnaitre les innombrables objets situés dans notre environnement. Nous apprenons très vite à décrire puis à reconnaître de nouveaux objets. Mieux encore, avec un seul oeil, capteur en deux dimensions, nous pouvons interpréter les trois dimensions dans lesquelles nous évoluons.

Il est évident que nous ne pouvons aujourd'hui demander à la machine des performances comparables : il s'avère nécessaire de sérieusement simplifier les problèmes et tout d'abord de limiter considérablement le nombre d'informations à traiter.

Une approche intéressante du modèle de vision humaine est signalée dans la bibliographie [7] p. 25 à 50, nous pouvons résumer cette étude en disant que la vision est le fait de deux types de capteurs : les bâtonnets et les cônes :

- les bâtonnets donnent une perception monochromatique, ils sont très sensibles,
- les cônes, de trois types fondamentaux (α, β, γ) permettent une perception des couleurs mais sont d'une beaucoup moins bonne sensibilité.



Beaucoup d'animaux sont démunis de ce deuxième type de capteurs et ne perçoivent l'environnement que par l'intensité du rayonnement lumineux (luminance). D'autre part, l'homme a une vision nocturne achromatique (à cause de la mauvaise sensibilité des cônes).

Ces considérations ont amené beaucoup de chercheurs à négliger la perception des couleurs. L'expérience montre que, généralement, cette notion apporte une information complémentaire plutôt que principale.

Nous allons maintenant décrire des capteurs sensibles aux mêmes types d'informations que celles perçues par l'oeil, voire même à d'autres types d'informations que le cerveau doit recréer (vision en trois dimensions).

II.2. PERCEPTION EN DEUX DIMENSIONS

Une image bidimensionnelle est une surface (virtuelle ou réelle) qui, en chacun de ses points (X, Y) se voit attribuer un vecteur $\overrightarrow{g(x, y)}$ appelé perception visuelle en ce point.

Un capteur d'images sera donc un appareil transmettant une ou plusieurs grandeurs (généralement électriques) proportionnelles aux composantes du vecteur $\overrightarrow{g(x, y)}$, balayé sur toute l'image.

Le capteur d'images le plus connu est la caméra de télévision :

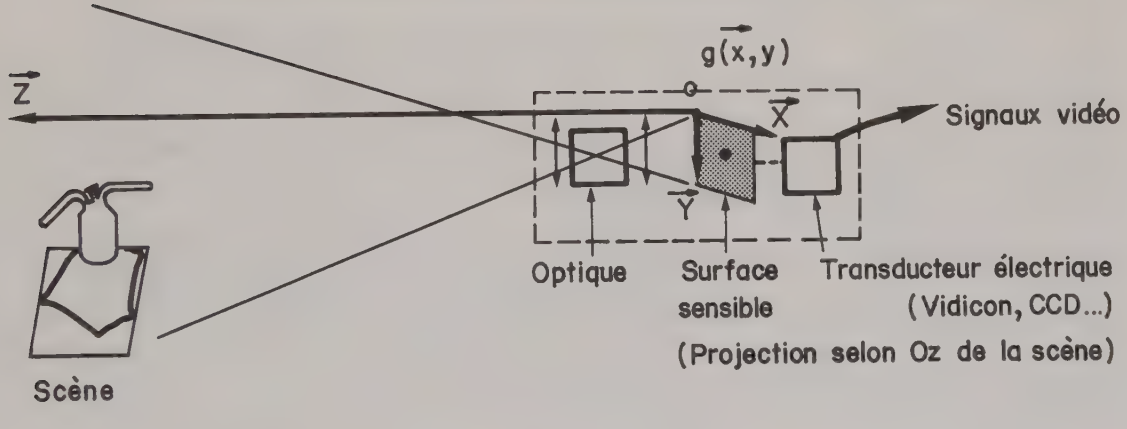


Figure 3.

en général, $\overrightarrow{g(X, Y)}$ se réduit à :

- une composante (caméra Noir et Blanc) proportionnelle à la luminance $\overrightarrow{g(x, y)} = \mathcal{J}(x,y)$

- trois composantes (caméra couleur) proportionnelles aux intensités de trois couleurs de base =

$$g(x,y) = \begin{pmatrix} \eta_R(x,y) \\ \eta_V(x,y) \\ \eta_B(x,y) \end{pmatrix}$$

Définition

On définit le grain ou pixel (de l'anglais picture cell) par le point le plus petit que peut distinguer le capteur, soit $(\Delta x, \Delta y)$ les dimensions de ce grain.

On peut se demander combien de grains sont nécessaires pour obtenir une "bonne" image :

- l'oeil humain a une définition d'environ 100 millions de grains,
- une image de cinéma - 4 000 000 grains
- une image de télévision - 600 000 grains
- une image de magnétoscope - < 100 000 grains.

En robotique, on utilise généralement des définitions allant de 128 x 128 ≠ 16 000 grains à 512 x 512 ≠ 250 000 grains (mais 256 x 256 ≠ 64 000 grains semble la définition la plus souvent employée).

D'autre part, on se limite presque toujours à l'analyse monochromatique (caméra Noir et Blanc) l'intensité $\eta(x,y)$ peut être discrétisée en un nombre de niveaux variant de deux (images binaires) à 256.

II.3. PERCEPTION EN TROIS DIMENSIONS

Un image tridimensionnelle est une surface (virtuelle ou réelle) qui, en chacun de ses points (x,y) se voit attribuer une valeur $Z(x,y)$ appelée relief, ainsi qu'un vecteur $\overrightarrow{g(x,y)}$ représentant la perception visuelle en ce point. $\overrightarrow{g(x,y)}$ est le plus souvent réduit à une valeur proportionnelle à la luminance en ce point.

II.3.1. Pseudo 3D

Dans la technique pseudo 3D, on cherche à retrouver visuellement des informations sur le relief à partir d'une image en deux dimensions [35].

Il s'agit, par analogie avec le borgne évoluant dans l'espace, de recréer l'information sur la profondeur par une interprétation assez élaborée des formes perçues en deux dimensions (l'objet B est devant l'objet A, ...).

On utilise, pour tenter de résoudre ce problème, des notions d'intelligence artificielle. Pour le moment, cette technique est hésitante en raison de la grande difficulté du problème. Il sera, de toute façon, impossible d'obtenir une image 3D telle que nous l'avons définie par cette méthode sans l'apport d'informations dimensionnelles supplémentaires telles que, par exemple, la taille réelle des objets perçus.

II.3.2. Technique K-2D

Dans la technique K-2D, on cherche à retrouver visuellement des informations sur le relief de la scène à partir de plusieurs images 2D, non identiques de celle-ci.

Il s'agit de retrouver dans plusieurs images la localisation des mêmes points physiques de la scène. La mise en correspondance de ces points permet de déduire une indication sur le relief [38].

Cette manière de faire trouve son origine dans le système visuel humain binoculaire. Cette technique (appelée stéréovision) a été expérimentée avec succès dans des reliefs peu tourmentés (par exemple, pour effectuer des relevés de lignes de niveau en cartographie) mais dans le cas de scènes complexes, il n'a pas été possible d'obtenir de résultats probants à ce jour à cause des problèmes liés à ce type de vision :

- il n'est pas toujours facile de trouver la correspondance entre les deux images. Cette correspondance est toujours longue à calculer (corrélation spatiale),

- dans certains cas, cette mise en correspondance est impossible (occlusion pour une caméra).

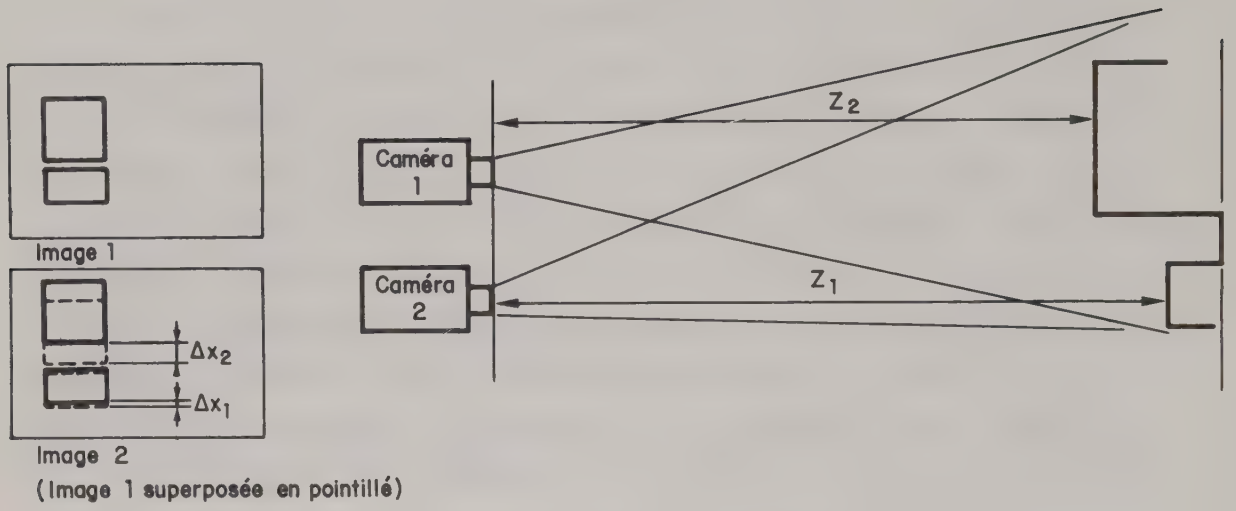


Figure 4 _ Principe de la stéréovision (déduire Z_i de Δx_i).

II.3.3. Capteurs trois dimensions

En général, il s'aqit d'ajouter à une caméra (donnant une image en 2D : g(x,y)), un télémètre qui donne l'information Z(x,y).

Idéalement, il ne doit pas exister d'interaction entre le télémètre et le percepteur visuel g(x,y).

Autrement dit, le télémètre ne doit pas perturber la mesure de g(x,y) : ce qui exclut a priori toute technique basée sur un système optique actif (triangulation par laser) au profit des systèmes agissant dans d'autres domaines (télémètre à radar ultra-sonique, radar hertzien).

En fait, si l'on parvient à obtenir une information suffisamment précise sur le relief, les données inhérentes à la vison perdent alors beaucoup d'intérêt. C'est pourquoi on utilise souvent la caméra non plus pour l'information visuelle g(x,y) de la scène, mais uniquement comme capteur associé au télémètre.

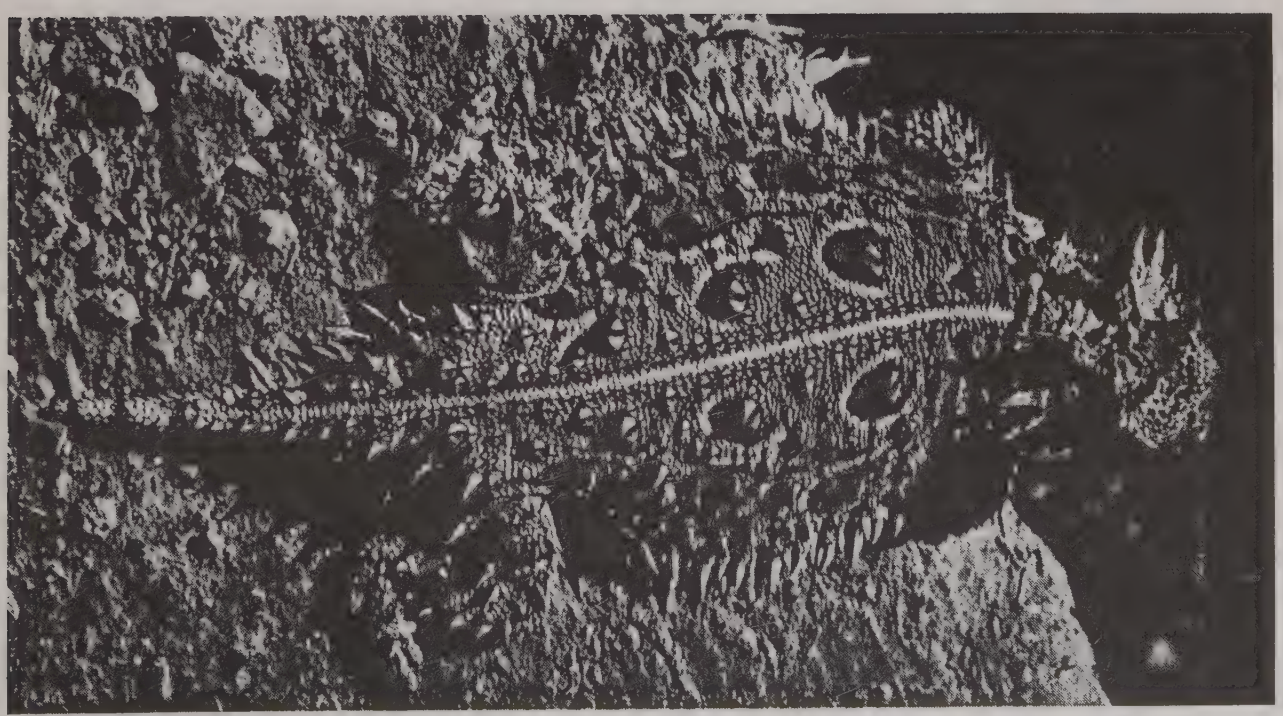
On ne peut plus alors parler d'image tridimensionnelle, mais plutôt d'une représentation en trois dimensions de l'espace sous la forme Z = f(X,Y).

Nous avons expérimenté un tel type de capteurs : nous décrivons en annexe son principe, ainsi qu'un projet d'utilisation de nos algorithmes travaillant sur les informations délivrées par un capteur de ce genre mais plus élaboré.

Plusieurs laboratoires ont développé des capteurs 3D déjà opérationnels : l'INRIA (Rocquencourt), le LETI (Grenoble), Hewlet Packard, S.R.I. [36], ...

II.4. EXTRACTION DE CONTOUR DESCRIPTEUR

II.4.1. Le problème



L'extraction d'informations pertinentes sur une image est très étroitement liée à la complexité de la scène à analyser.

Pour reconnaitre une forme par rapport à notre représentation du monde, nous ferons tantôt appel :

* au contexte (historique des informations perçues)
* aux différentes perceptions que nous avons de la forme :

    couleur = chercher des fraises
    brillance = vision nocturne
    texture = chercher de la mousse dans du gazon.

mais plus généralement, la première démarche que nous effectuons consiste à chercher les limites de cette forme dans la mesure où ces limites sont perceptibles. Par ailleurs, il semblerait que l'oeil ne soit sensible qu'aux variations :

- temporelles = l'apparition subite d'une forme dans notre champ visuel attire instantanément notre attention ; inversement, si nous réalisons un montage optique afin qu'une forme suive fidèlement tous les mouvements de l'oeil, l'observateur aura l'impression que celle-ci disparaît en quelques secondes.
- spatiales = l'oeil est naturellement attiré par les transitions brutales de niveau : par exemple, une transition entre deux zones homogènes sera naturellement accentuée par l'oeil. C'est ainsi que nous voyons sur l'image de la figure 6 :

    . le bord A plus clair que B
    . le bord C plus foncé que D.

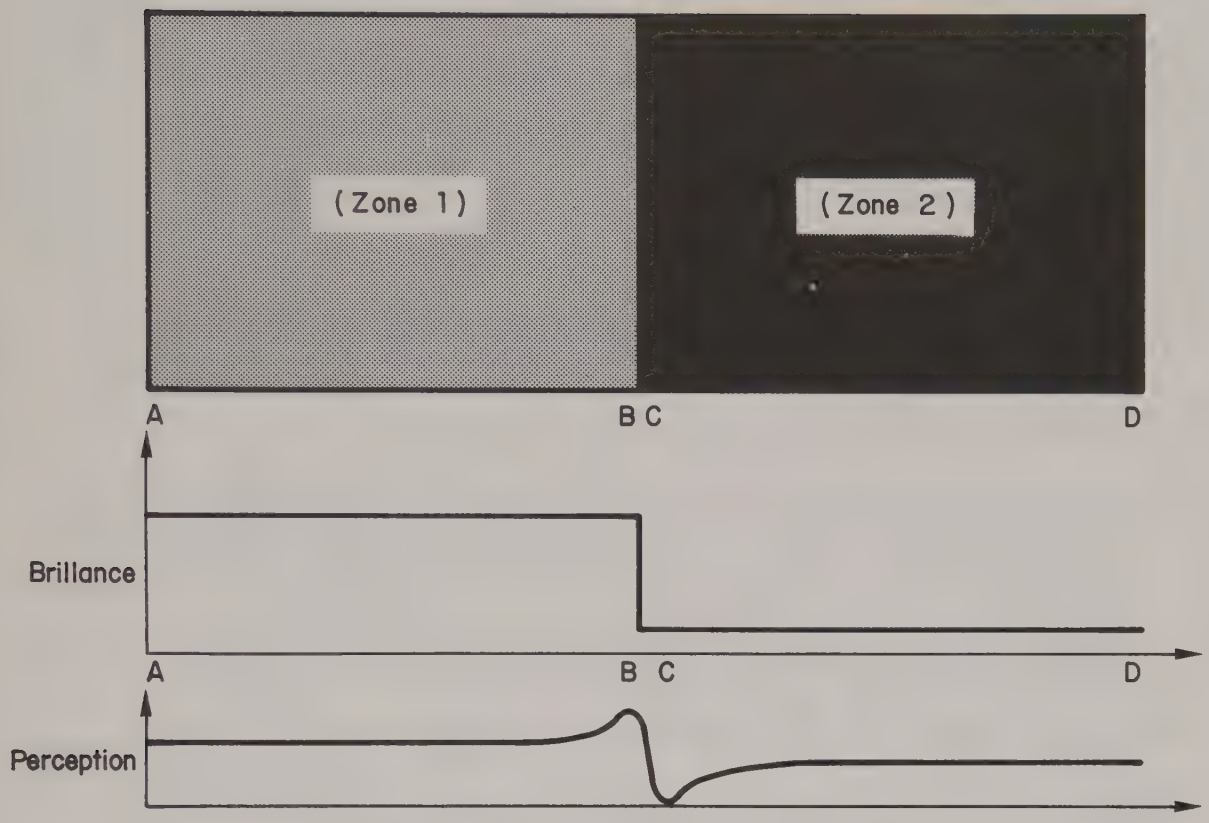


Figure 6 _ Un exemple de l'accentuation naturelle des bords par l'œil.

L'aspect différentiel de la vision humaine est très important : il permet de percevoir quasi-instantanément certains dangers immédiats (projectile en mouvement, animal chargeant, etc...)

De telles observations ont orienté beaucoup de recherches vers l'analyse de la notion de contour. Nous pouvons

définir, d'une façon générale, un contour comme une transition entre deux zones perçues par l'oeil comme homogènes.

Cette homogénéïté peut être due à une constante de brillance, de couleur ou de texture.

Il semble, cependant, que la brillance apporte la plupart de l'information dans la perception visuelle (très peu d'animaux voient les couleurs et la texture apporte plutôt une information complémentaire dans la majorité des observations).

II.4.2.

Nous limiterons, dans la suite de l'exposé, la définition de contour comme étant la ligne correspondant aux fortes transitions de niveau d'intensité.

La notion de transition peut s'exprimer mathématiquement en chaque point de l'image par le gradient d'intensité calculé sur une fenêtre de taille spécifiée.

Ainsi, dans le cas général :

$$\vec{G} = \overrightarrow{\text{grad}}\, \mathcal{J} = \frac{\partial \mathcal{J}}{\partial \ell}\ \vec{\ell}^{\,o}$$

et, dans un repère cartésien lié à l'image :

$$\vec{G} = \frac{\partial \mathcal{J}}{\partial x}\ \vec{i} + \frac{\partial \mathcal{J}}{\partial y}\ \vec{j}$$

Nous pouvons donc définir le contour comme l'ensemble de lignes à gradient localement maximal. Remarquons qu'une telle définition permet d'obtenir une orientation pour chaque ligne de contour, par référence à l'orientation du gradient (en d'autres termes, nous définissons le contour par les lignes d'isoscalaires nuls du Laplacien correspondant au gradient maximal, auquel on attribue une orientation arbitraire).

II.5. LES ALGORITHMES TRADITIONNELS

Nous n'avons pas la prétention de décrire tous les algorithmes conventionnels de détection de contour, mais simplement de rappeler les principes des opérateurs les plus connus que nous avons expérimentés.

II.5.1. Seuillage

Si l'intensité mesurée sur la forme à analyser se distingue nettement du fond ; il suffira de faire un seuillage sur l'image :

$$\mathcal{F}(\mathcal{J}) = 1 \quad \text{si } \mathcal{J} \geq \mathcal{J}_S$$

$$\mathcal{F}(\mathcal{J}) = 0 \quad \text{sinon}$$

Le seuil $\mathcal{J}_S$ pouvant :

- soit être fixé une fois pour toutes
- soit être fixé après examen de l'histogramme des intensités : si celui-ci présente deux sommets nettement séparés par une vallée et si chacun des sommets correspond respectivement aux points de l'objet et du fond, l'histogramme est dit bi-modal et le seuil $\mathcal{J}_S$ sera déterminé par la valeur minimale du col.

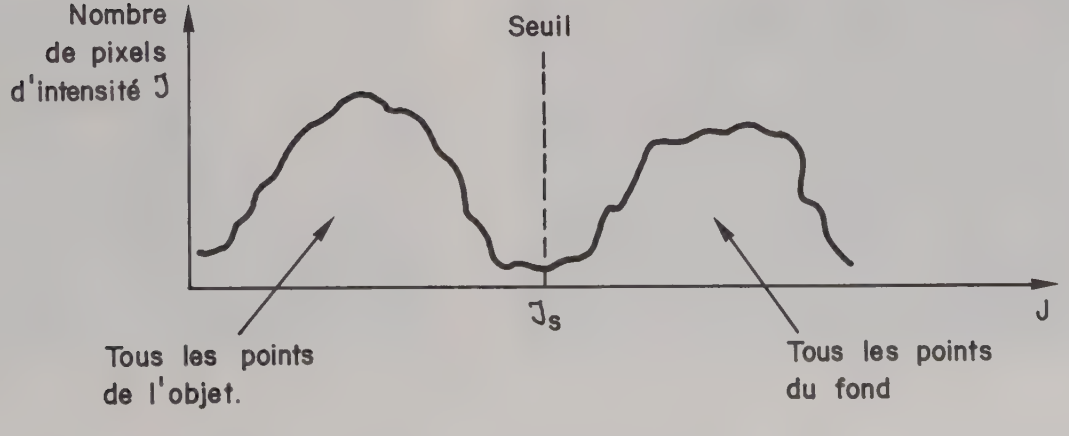


Figure 7 _ Un exemple d'histogramme bi-modal.

Image fortement contrastée

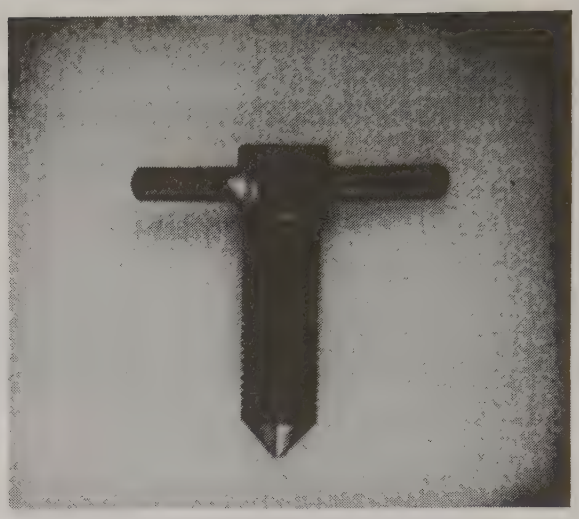
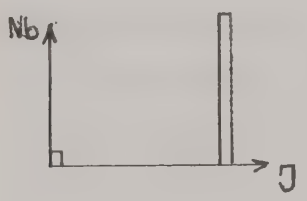
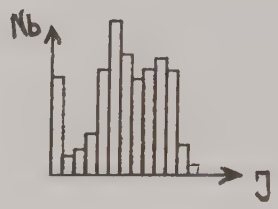


Image moyennement contrastée

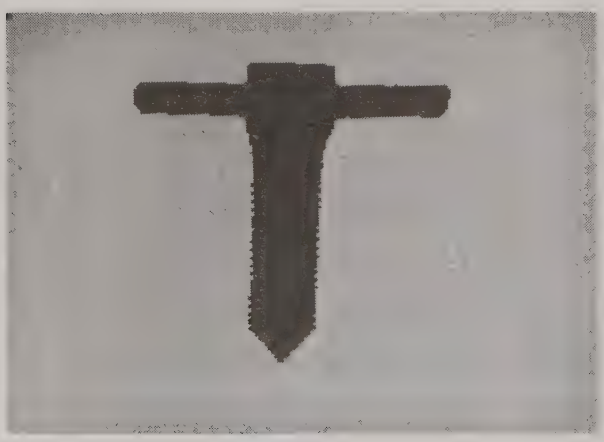


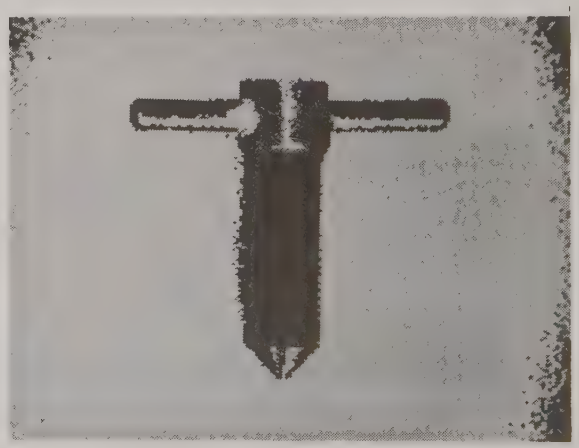
Histogramme bi-modal



où mettre le seuil ?

Histogramme multi-modal





Résultats après seuillage

FIGURE 8 : SEUILLAGE

La figure 8 montre que ce traitement ne peut convenir que dans le cas d'images fortement contrastées (par exemple : objet placé sur une plaque translucide lumineuse), dans le cas d'images moyennement contrastées, l'histogramme n'est plus bimodal. On peut encore utiliser cette méthode en divisant l'image jusqu'à obtenir un histogramme bimodal pour chaque sous-image. Mais, il se pose alors le problème du recollage des sous-images : il peut, en effet, s'avérer difficile de faire correspondre les contours trouvés dans chaque sous-image.

Le seuillage est une opération très simple, très rapide, facile à câbler mais reste très limité quant à ses performances et ne peut pas être utilisé dans des conditions médiocres d'éclairage pour de l'analyse de scènes.

La plupart des systèmes de vision développés aujourd'hui en robotique utilisent la technique du seuillage et nécessitent donc une prise de vue parfaite qui n'est pas toujours réalisable en milieu industriel.

II.5.2. Transformées de Fourier

L'opérateur linéaire le plus classique en traitement d'images est le filtre en fréquences spatiales [7], [8].

$$\mathcal{F}(\omega_x,\omega_i) = \iint \mathcal{J}(x,y)\ e^{-i(\omega_x x + \omega_y y)}\ dx\ dy$$

$\mathcal{J}(x,y)$ est l'intensité du point (x,y)

ω_x est la pulsation spatiale selon ox

ω_y est la pulsation spatiale selon oy

Les brutales variations d'intensité se traduisent par de hautes fréquences spatiales. Ainsi on peut accentuer les contours en réalisant un filtre passe haut $\mathcal{H}(\omega_x,\omega_y)$

$$\mathcal{G}(\omega_x,\omega_y) = \mathcal{F}(\omega_x,\omega_y).\mathcal{H}(\omega_x,\omega_y).$$

Ce qui se traduit, dans le domaine de l'image, par :

$g(x,y) = \mathcal{J}(x,y) \otimes h(x,y)$ $\otimes$ produit de convolution.

On obtient des contours accentués et l'élimination des parties homogènes, mais ce filtrage fait aussi ressortir les parasites et le bruit indésirable à cause de leur importante énergie dans les hautes fréquences.

Plusieurs filtres optimisés ont été mis au point [10], [11].

De telles opérations sont relativement coûteuses en temps de calcul. Elles peuvent être câblées mais nécessitent un matériel relativement cher (structures pipe-line, array-processeurs).

II.5.3. Algorithme de convolution

Nous avons défini (II.1) le contour à l'aide de la notion de Laplacien.

Beaucoup de chercheurs ont eu l'idée de réaliser une approximation de cet opérateur par une convolution sur l'image avec une fenêtre dont les paramètres sont calculés en fonction de la réponse désirée.

Considérons, par exemple, une ligne de contour idéale.

Digitalisation :

ligne I-1 =	2	2	2	4	8	12	14	14	14	14
ligne I =	2	2	2	4	8	12	14	14	14	14
ligne I+1 =	2	2	2	4	8	12	14	14	14	14

Résultat de la convolution par $\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$

0 -6 -6 +6 +6 0 0

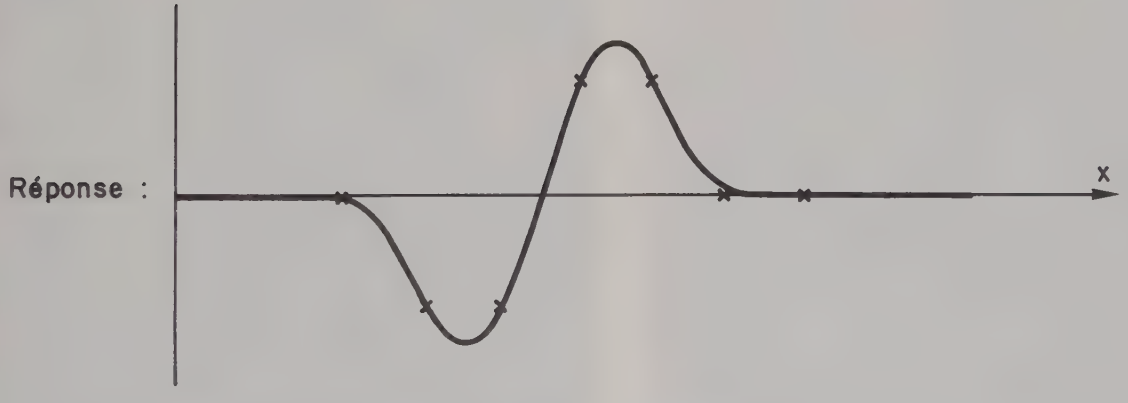


Figure 9 _ Convolution par un "Masque de Laplacien".

La figure 9 montre que la convolution par un masque judicieusement choisi donne une bonne estimation du Laplacien calculé dans un voisinage de 8 points.

De nombreuses fenêtres de tailles plus ou moins conséquentes ont été proposées : pour notre part, nous exposons quelques approximations du Laplacien par des convolutions 3x3 (figure 10).

Nous remarquons la grande sensibilité au bruit de ces algorithmes, due au caractère ponctuel des masques (le terme central du masque a obligatoirement une valeur élevée, ce qui favorise le tracé des parasites ponctuels).

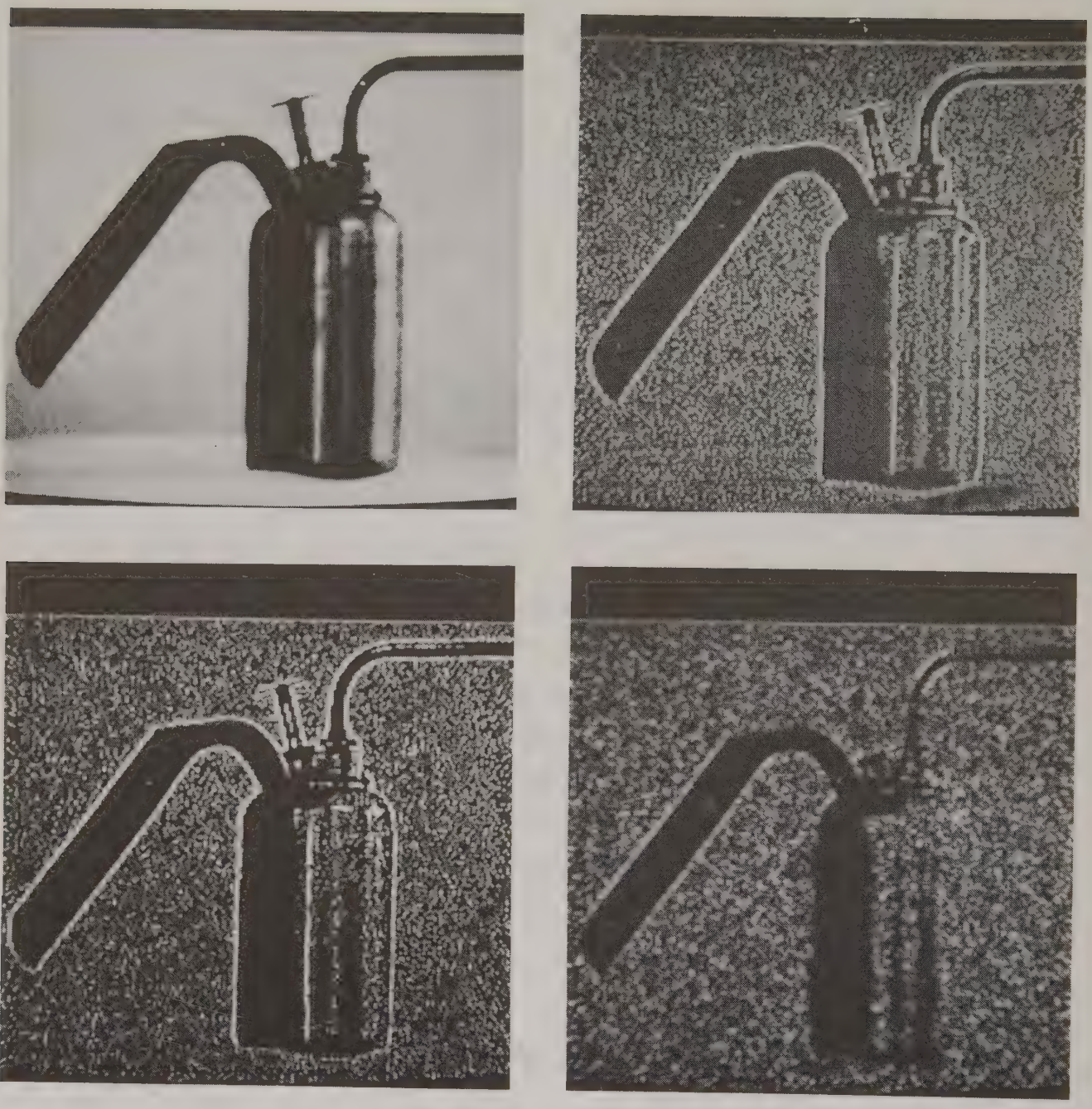


FIGURE 10 : ALGORITHMES DE CONVOLUTION

a) Image originale

b) Détecteur "pointu"

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

c) Détecteur "large"

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$$

d) Détecteur "à bord évasés"

$$\begin{matrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{matrix}$$

Cet algorithme peut être câblé assez aisément.

II.5.4. Algorithmes définis à partir du gradient

Ces algorithmes possèdent la propriété de permettre le calcul des directions privilégiées du contour contrairement à l'algorithme du Laplacien qui perd cette information.

Revenons à la définition cartésienne du gradient en un point P(x,y) :

$$\vec{G} = \frac{\partial \mathcal{J}}{\partial x} \vec{i} + \frac{\partial \mathcal{J}}{\partial y} \vec{j}$$

Une bonne estimation des composantes du gradient en P s'obtient en multipliant matriciellement une fenêtre de taille m x n par le voisinage de P sur l'image limitée à la taille m x n.

On approche $\frac{\partial \mathcal{J}}{\partial x}$ par un détecteur de lignes verticales dont voici quelques exemples :

$$G_x = A_x \mathcal{J}(x,y) \text{ avec}$$

$$A_{x_1} = \begin{bmatrix} -1 & +1 \end{bmatrix}, \quad A_{x_2} = \begin{pmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{pmatrix}, \quad A_{x_3} = \begin{pmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{pmatrix},$$

$$A_{x_4} = \begin{pmatrix} -1 & 0 & 0 & +1 \\ -2 & -1 & +1 & +2 \\ -2 & -1 & +1 & +2 \\ -1 & 0 & 0 & +1 \end{pmatrix}$$

$\frac{\partial \mathcal{J}}{\partial y}$ est estimé par un détecteur de lignes horizontales :

$$G_Y = A_Y \mathcal{J}(x,y) \text{ avec}$$

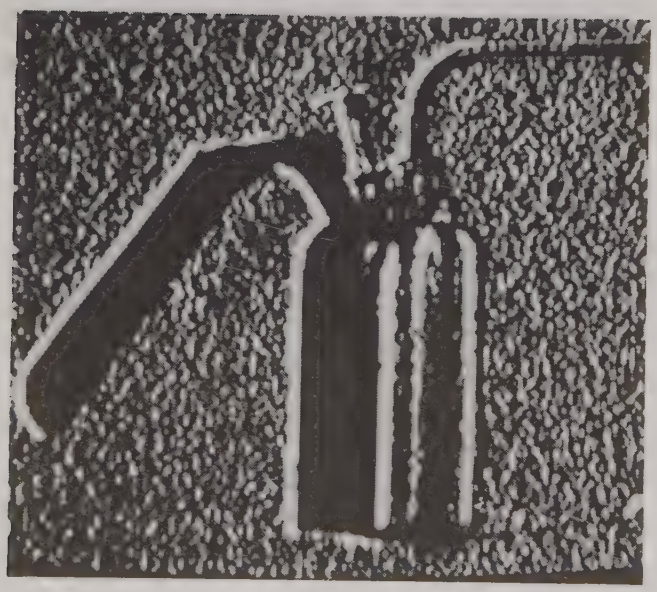
$$A_{Y_1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad A_{Y_2} = \begin{pmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, \quad A_{Y_3} = \begin{pmatrix} +1 & +1 & +1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix}$$

$$A_{Y_4} = \begin{pmatrix} +1 & +2 & +2 & +1 \\ 0 & +1 & +1 & 0 \\ 0 & -1 & -1 & 0 \\ -1 & -2 & -2 & -1 \end{pmatrix}$$

On peut calculer alors :

- le module du gradient = $\| \vec{G} \| = \sqrt{G_x^2 + G_Y^2}$

- l'argument du gradient = Arg $\vec{G}$ = Arc tg $\frac{G_Y}{G_x}$



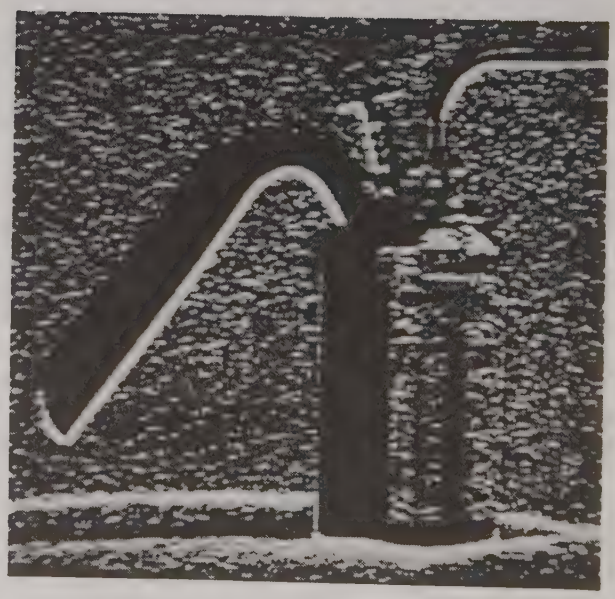


FIGURE 11 : Composantes du gradient calculées avec A_{x_2} et A_{Y_2}

a) Image originale
b) Composante selon O_x
c) Composante selon O_Y

La recherche de contours à partir d'algorithmes basés sur cette notion consiste à :

- calculer le gradient en tout point de l'image (phase d'accentuation des contours)
- chercher à relier les maxima d'amplitude en tenant compte de la direction du gradient [13] (phase de détection des lignes de contour et recherche de connexité).

La première phase exige généralement le plus de soins dans l'optimisation, car c'est ce prétraitement qui définit en grande partie les performances globales du système d'analyse.

- Plusieurs algorithmes ont simplifié les calculs afin de rendre le traitement plus rapide, notamment :

Sobel = approximation des composantes du gradient

$$\vec{G} = (A_{X_2} \rceil, A_{Y_2} \rceil)$$

approximation du module

$$\|\vec{G}\| = G_X + G_Y$$

Robert = approximation des composantes du gradient

$$\vec{G} = (A_{X_1} \rceil, A_{Y_1} \rceil)$$

approximation du module

$$\|\vec{G}\| = G_X + G_Y$$

Prewitt = approximation des composantes du gradient

$$\vec{G} = (A_{X_3} \rceil, A_{Y_3} \rceil)$$

calcul réel du module

$$\|\vec{G}\|^2 = G_X^2 + G_Y^2$$

La figure 12 montre les résultats donnés par ces algorithmes.

Nous remarquons que l'algorithme de Sobel accentue assez bien tous les contours, même lorsqu'ils sont peu prononcés.

L'algorithme de Robert est plus sensible au bruit et moins performant pour les faibles contrastes. Cependant, il est beaucoup plus rapide, car demande quatre fois moins de calculs pour le réaliser.

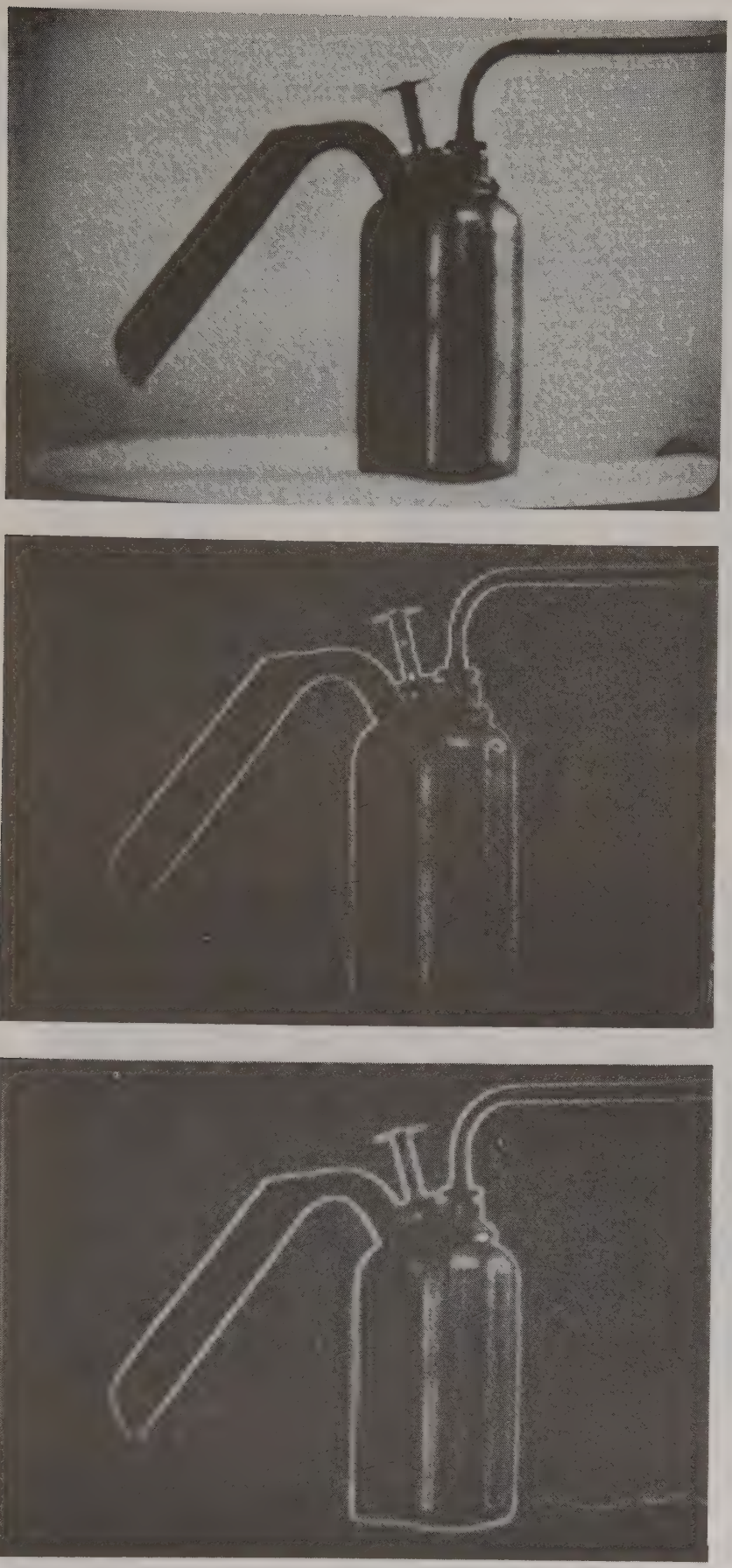


FIGURE 12 : Algorithmes d'extraction de contour

1) Image originale
2) Algorithme de Robert
3) Algorithme de Sobel

• Une autre méthode d'accentuation des contours consiste à créer des masques privilégiant une direction particulière du gradient. C'est le cas notamment de :

- Kirsch : 8 masques correspondants aux 8 directions possibles de voisinage :

```
 3  3 -5        3 -5 -5        -5 -5 -5
 3  0 -5        3  0 -5         3  0  3   (. . .)
 3  3 -5        3  3  3         3  3  3
    ←              ↙              ↓
détecteur
```

- Compass : même idée que Kirsch mais avec d'autres coefficients :

```
 1  1 -1        1 -1 -1        -1 -1 -1
 1 -2 -1        1 -2 -1         1 -2  1  (. . .)
 1  1 -1        1  1  1         1  1  1
    ←              ↙              ↓
détecteur
```

La figure 13 montre que les contours sont remarquablement bien accentués. Cependant, de tels algorithmes consomment beaucoup de temps de calcul : il faut effectuer 8 convolutions sur l'image pour obtenir ce résultat !...

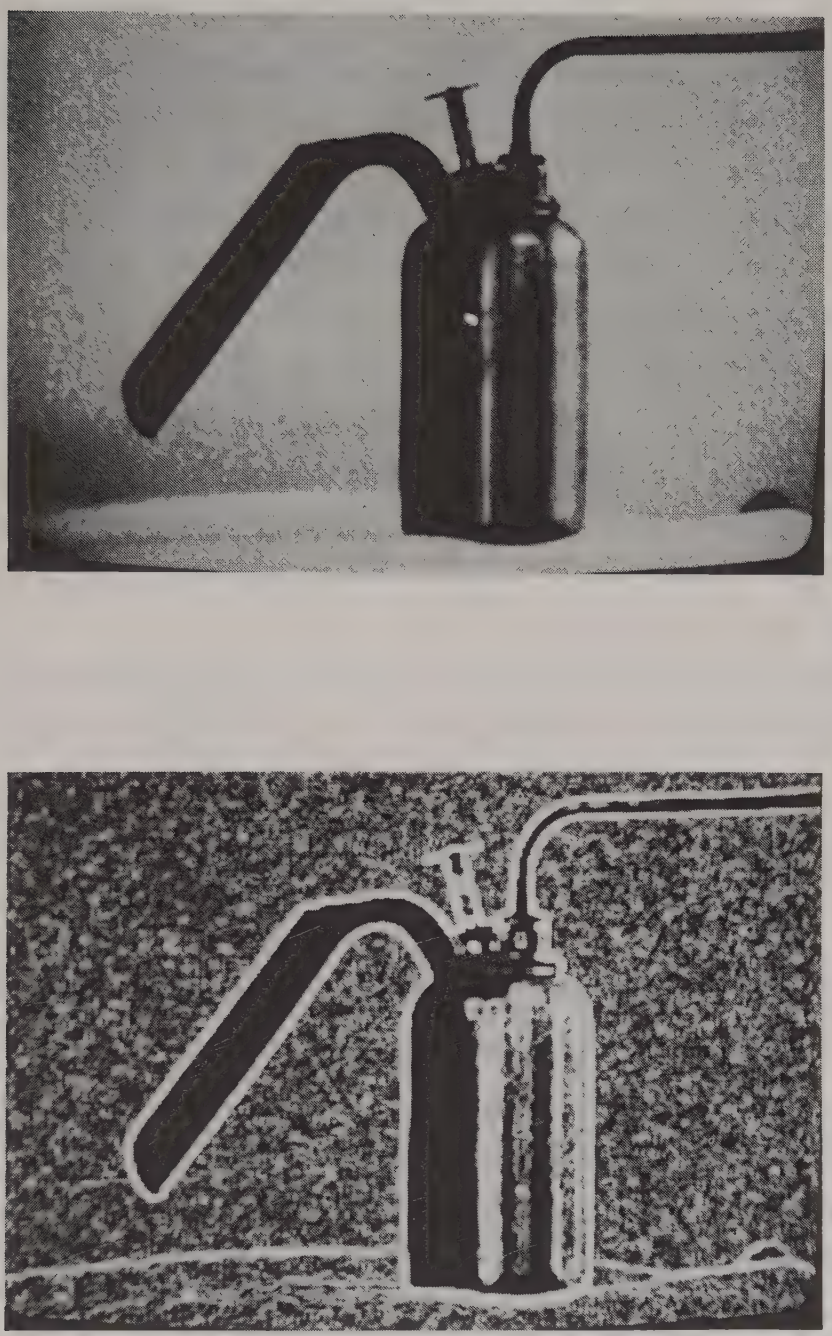


FIGURE 13 : Algorithme de Kirsch
a) Image originale
b) Kirsch

Résultats :

Tous les algorithmes exposés dans ce paragraphe possèdent l'avantage d'être plus proches de la notion de gradient, et surtout de conserver la notion d'orientation.

Ils sont cependant un peu plus longs (2 ou plusieurs convolutions) mais présentent l'avantage de pouvoir être câblés, car ils effectuent le même traitement en chaque point de l'image.

II.5.5. Autres méthodes

Nous avons prévenu le lecteur qu'il ne trouvera pas ici une revue de toutes les méthodes de détection de contour à cause de la trop grande variété d'approches de ce problème qui a fait l'objet de nombreuses publications. Nous le renverrons donc à la bibliographie, en signalant toutefois :

- Les méthodes de recherches de coïncidences de contour Hueckel [12], Pavlidis [13] [14], Hough [18].
- Les méthodes par recherches de régions de propriétés similaires (région growing, clusterring, split and merge, quadtrees) [15], [16], [17], [8].
- Les méthodes statistiques [19], l'analyse de textures [20] .

II.6. LE CODAGE DU CONTOUR

Considérons maintenant une image digitalisée spatialement en une matrice M x N.

Plusieurs techniques peuvent être utilisées pour coder les lignes de contour.

II.6.1. Le codage par coordonnées : on peut stocker dans une mémoire les coordonnées cartésiennes de chaque point de contour. La méthode se révèle particulièrement peu efficace dès que l'image présente des contours continus (il est alors nécessaire d'enregistrer beaucoup d'informations pour des points adjacents).

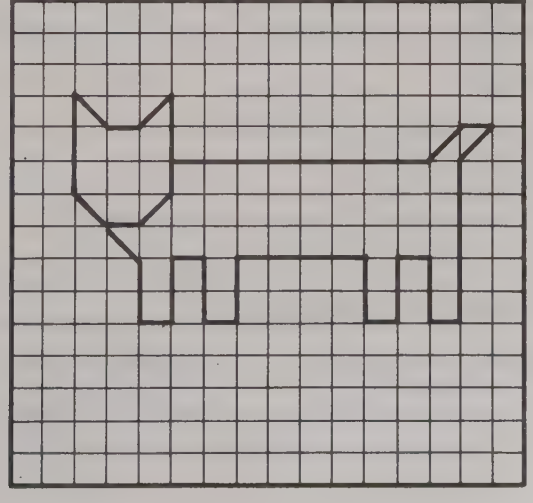
II.6.2. Le codage par marqueurs : on peut associer à l'image une matrice N x M binaire. L'état de chaque cellule indique la présence (1) ou l'absence (0) de contour. Un tel codage présente l'avantage de conserver la notion spatiale du contour (on peut rapidement savoir si un point de contour est proche d'un autre).

II.6.3. Le codage par vecteurs de FREEMAN : ce codage est très intéressant pour l'encombrement mémoire si les contours sont continus. La méthode consiste à coder les déplacements élémentaires que l'on effectue en suivant la ligne de contours.

Dans notre représentation matricielle, il n'y a que 8 possibilités pour passer d'un point à un autre par adjacente (côté commun ou coin commun). Codons ces directions possibles (de 0 à 7), nous obtenons une représentation compacte de n'importe quelle ligne.

Il suffit donc d'enregistrer les coordonnées des départs de chaque ligne ainsi que les chaines de Freeman associées pour avoir tous les renseignements nécessaires à sa restitution. Une telle représentation est relative.

Supposons, par exemple, que notre détecteur de contour nous ait donné deux lignes dans une image 16 x 16 :



Nous coderons :

Pt de départ (0) = 2,3
chaine (0) = début,7,0,1,6,6,6,5,4,3,2,2,2, Fin
Pt de départ (1) = 5,5
chaine (1) = début,0,0,0,0,0,0,0,0,1,0,5,6,6,6,6,6,
4,2,2,4,6,6,4,2,2,4,4,4,4,6,6,4,2,2,4,
6,6,4,2,2,3, Fin.

Un tel codage est très intéressant car, il fait apparaître le caractère vectoriel de la ligne par l'orientation de ses vecteurs élémentaires. Par ailleurs, de nombreux travaux en traitement des images ont été réalisés en utilisant ce principe ; de nombreux algorithmes sont disponibles dans la littérature. Cependant, les propriétés spatiales du contour ne sont pas immédiates (pour savoir si un contour passe par un point, il est nécessaire de "dérouler" la chaîne et d'examiner successivement tous les points de la ligne).

II.6.4. Compression

L'encombrement mémoire dépendra de la longueur et du nombre de lignes à coder. Si on pose :

$$2^{n+1} < N \leq 2^{n}$$

$$2^{m-1} < M \leq 2^{m}$$

ℓ la longueur totale des lignes en nombre de points,
p le nombre de lignes.

La taille mémoire consommée par chaque codage sera en nombre de bits :

- coordonnées = $\ell(n+m)$
- marqueurs = n m
- Freeman = $p(n+m) + 3\ell$

Dans notre exemple précédent :
- coordonnées : 53 x 8 = 424 bits
- marqueurs : 16 x 16 = 256 bits
- Freeman : 16 + 3 x 53 = 175 bits

CHAPITRE III : NOTRE METHODE : SUIVI DYNAMIQUE DE CONTOUR

III.1. Principe

Les méthodes exposées précédemment impliquent l'examen de tous les points de l'image. Dans le cas d'un traitement séquentiel, ce calcul systématique allonge considérablement le temps d'exécution. Or, il est bien évident que l'application d'un tel alogrithme dans une zone homogène sera une perte de temps totale. Inversement, si nous avons détecté plusieurs points correspondant à une ligne de contour, il est probable que nous en détecterons d'autres par continuité.

D'où l'idée de faire intervenir cette notion de continuité d'un contour dans l'algorithme. Autrement dit, dès que l'on trouve l'amorce d'une ligne de contour, nous allons essayer de poursuivre cette ligne en essayant de la prolonger par continuité. Un tel algorithme présente, de surcroît, l'avantage d'indiquer implicitement la connexité du contour.

Voici un exemple simple utilisant cette démarche : le gradient dynamique.

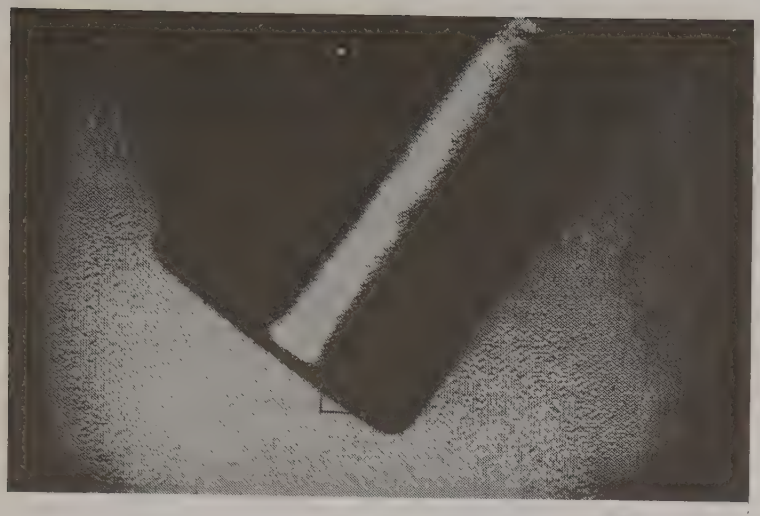
Cet algorithme prend la représentation simplifiée de Sobel (voir II.5.4.) pour la détermination du gradient. La stratégie consiste à rechercher un point à gradient maximal en explorant un chemin sur l'image (phase dite de "balayage"). Soit I(x,y) un point de l'image satisfaisant cette condition, il est considéré comme "amorce" d'un contour.

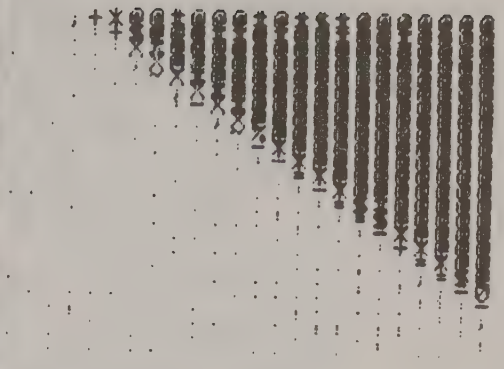
L'algorithme va alors se placer au point J(x',y') adjacent à I dans la direction perpendiculaire au gradient en I. Si l'amplitude du gradient en J est supérieure à un seuil prédéfini, alors J est pris comme nouveau point de contour à partir duquel recommence une nouvelle itération. La "poursuite" se continue à moins que :
- l'amplitude du gradient devienne insuffisante,
- l'on rencontre un point déjà traité.

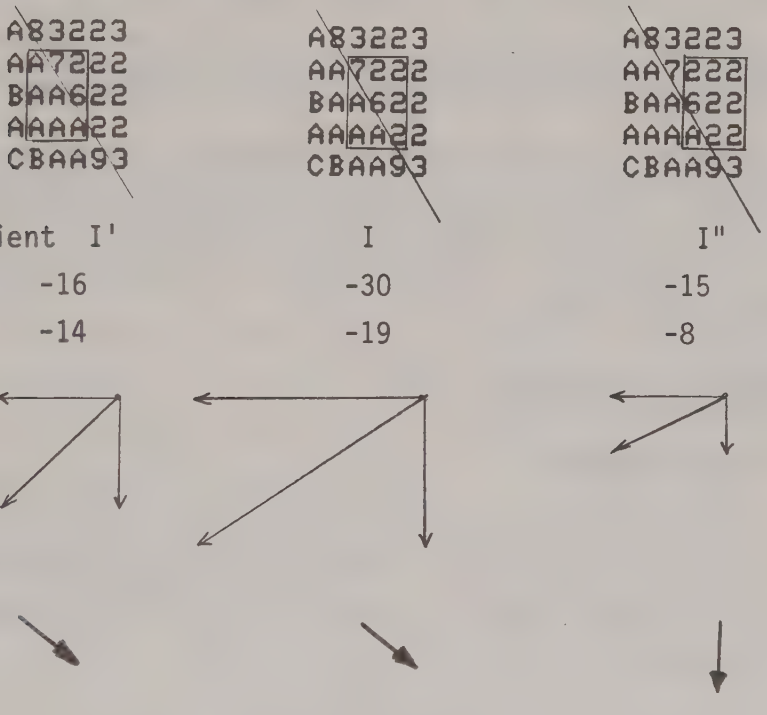
Si une de ces conditions est rencontrée, la poursuite est abandonnée et l'on retourne au balayage.

L'algorithme du gradient dynamique permet de trouver rapidement un contour dans une image. Il est assez efficace car il tend lui-même à corriger ses écarts par rapport au contour :

Prenons, parexemple, une vue réelle d'un élément de contour de notre burette d'huile et calculons le gradient :
- en un point I situé sur le gradient maximum,
- en deux points I' et I" adjacents à I (décalés donc par rapport au gradient maximum).







Composantes du gradient	I'	I	I"
G_x =	-16	-30	-15
G_y =	-14	-19	-8

Gradient:

Direction de poursuite:

Figure 16 : Le gradient dynamique est asservi à la poursuite du contour

Nous remarquons que l'algorithme tend à attribuer dans tous les cas le point J pour nouveau point de contour. Si l'image est de bonne qualité, l'algorithme se "raccroche" à la ligne de contour. En revanche, un tel algorithme devient vite inefficace avec le bruit ; le gradient représentant une dérivée, celle-ci se trouve très affectée par les parasites ponctuels, tant pour l'amplitude que pour la direction dans de tels cas, l'algorithme peut être dérouté et va abandonner la poursuite à cause de l'insuffisance du gradient. On dit que l'algorithme "décroche".

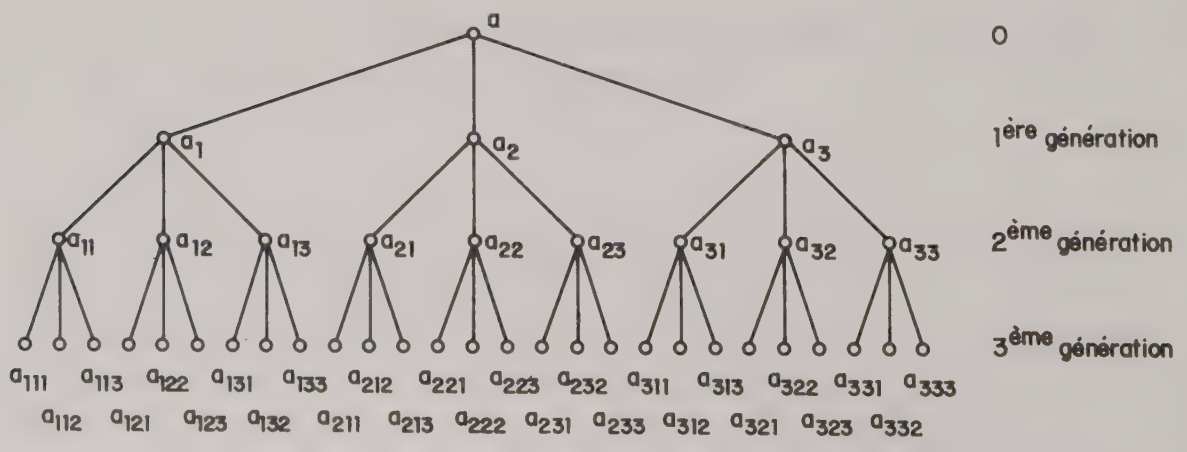
En conclusion, cette méthode est très performante car assez rapide (il n'est pas nécessaire de l'appliquer en tous les points de l'image) mais elle ne donne pas d'assez bons résultats dans le cas d'images peu contrastées, car elle devient très sensible au bruit et décroche facilement.

Nous avons donc décidé de concevoir un algorithme inspiré de cette méthode mais englobant des critères moins locaux afin de la rendre performante dans le cas d'images moyennement bruitées.

III.2. METHODE PROPOSEE

III.2.1. Arborescence

Considérons une structure arborescente à n générations constituée par des noeuds ayant chacun ℓ fils. Une telle structure a, lors de la $n^{ième}$ génération, ℓ^n enfants.



Parcourir ce graphe signifie aller du sommet (Ancêtre) vers une feuille (un élément de la dernière génération).

On peut attribuer à chaque branche (liaison entre un père et son fils) un nombre appelé coût de succession $c(a_i,a_{ij})$. Dans ce cas, le parcours du graphe coûtera la somme des coûts rencontrés lors du parcours, ce qui se calcule, pratiquement, en partant de la feuille visée et en additionnant les coûts jusqu'au sommet.

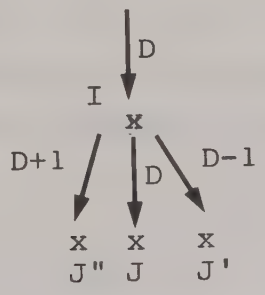
Trouver le parcours à coût minimal oblige nécessairement à explorer successivement tous les chemins possibles, ce qui peut être une stratégie longue, surtout si le nombre de générations est élevé.

III.2.2. Application d'une structure arborescente à l'extrêmité de contour

On appelle micro-ligne toute ligne reliant deux points adjacents. Supposons que nous ayons trouvé en un point I une micro-ligne de contour dans la direction D du rodage de FREEMAN (cf. II.6.3). On appelle successeurs de I les points J, J' et J" adjacents à I et situés dans les directions D, $(D-1)_*$ et $(D+1)_*$.

* signifie opération modulo 8.

Exemple : D = 2



Chaque point de contour a donc trois successeurs possibles. On définit ainsi une ligne de contour comme un parcours particulier dans une arborescence dont le sommet est un point de direction donnée et dont les branches sont les différentes successions générées, choisies arbitrairement.

Corollaire : une ligne de contour ne peut être constituée de micro-contours formant entre eux un angle de plus de 45° en valeur absolue.

Les lignes de contour seront donc "lissées". Nous pouvons ainsi représenter les lignes de contours générées à partir d'un sommet de direction O.

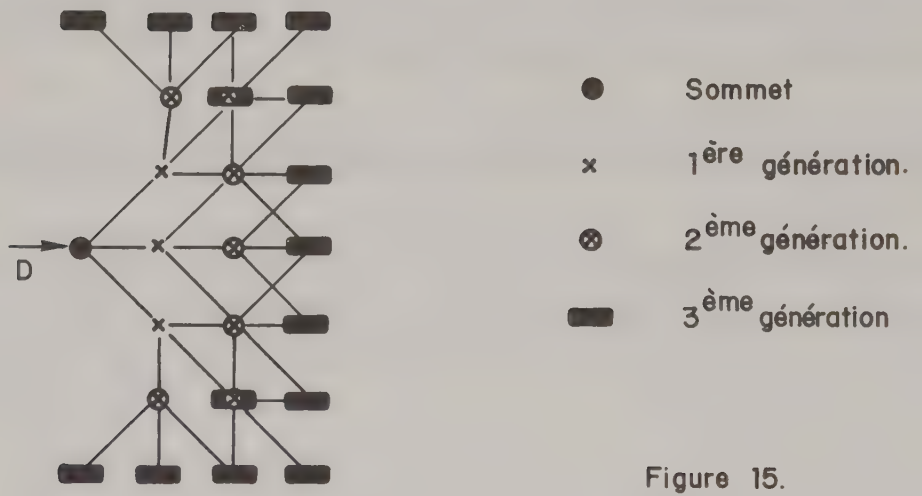


Figure 15.

Nous voyons qu'un même point peut être issu de générations différentes (on peut aller au point A de deux manières 0 1 0 ou 1 1.).

Dans le premier cas, nous avons traversé trois générations alors que dans le deuxième cas deux générations seulement ont été nécessaires.

L'idée de base de cet algorithme consiste à associer à chaque déplacement dans le graphe un coût de succession indiquant la qualité du contour suivi : plus le coût est faible et plus le micro-contour correspondant est supposé être de bonne qualité.

Dans de telles conditions, la détection de lignes de contour correspond à une recherche de parcours dans le graphe avec un coût minimal.

III.2.3. Détermination d'une fonction de coût

Nous avons vu qu'un contour était déterminé par les lignes à gradient localement maximal. Notre fonction de coût va faire intervenir cette notion à partir de la valeur de l'amplitude du gradient au point I générateur $G(\mathcal{J}_I(x,y))$.

$\mathcal{J}_I(x,y)$ est l'intensité au point $I(x,y)$.

Soit J le successeur étudié et $G(\mathcal{J}_J(x,y))$ le gradient en ce point.

Le coût sera proportionnel à la différence des gradients.

$$C_1 = A\left[G(\mathcal{J}_I(x,y)) - G(\mathcal{J}_J(x,y))\right]$$

Si toutefois $G(\mathcal{J}_I(x,y))$ est supérieur à une certaine valeur minimale G_M sinon :

$$C_1 = A\left[G_M - G(\mathcal{J}_J(x,y))\right].$$

Autrement dit, le coût sera plus important si on arrive dans une zone où le contraste diminue. Si celui-ci chute trop brutalement, (c'est à dire si C_1 devient supérieur à un seuil imposé), alors la fonction de coût se voit alourdie d'un terme "recentreur" faisant intervenir l'écart entre le niveau de gris en $J = \mathcal{J}_I(x,y)$ et une valeur moyennée de l'intensité du père $\mathcal{J}_I^*(x,y)$.

$$\mathcal{J}_I^*(x,y) = MI \text{ avec } M = \frac{1}{9}\begin{pmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{pmatrix}$$

$$C_2 = B\left[\mathcal{J}_J(x,y) - \mathcal{J}_I^*(x,y)\right]$$

Intérêt de C_2 : lorsque l'on se trouve dans une zone "perturbée", l'information du gradient n'est plus toujours valable. Ce terme tend à "recaler" la poursuite sur la ligne de transition du contour en utilisant seulement l'information niveau de gris de la transition.

D'autre part, la fonction de coût doit faire intervenir la direction en J du gradient : cette direction est-elle compatible avec l'arborescence ou non ? En pratique, si elle ne l'est pas, on donne une valeur maximale au coût afin qu'elle ne soit pas sélectionnée :

$C_3 = 0$ si $D_I - 1 \leq D_J \leq D_I + 1$

$C_3 = +\infty$ sinon

La fonction retenue sera donc :

$C = C_1 + C_3$ si $C_1 < C_{1\ MAX}$

$C = C_1 + C_2 + C_3$ sinon.

La ligne de contour optimale sera le parcours à coût minimal.

III.3. APPLICATION DE LA METHODE AU SUIVI DE LIGNES DE CONTOUR DANS UNE IMAGE : PRESENTATION DE L'OBJECTIF

III.3.1. Recherche de points de départ pour la poursuite

Si on suppose que la taille minimum de la forme à retrouver à une longueur L et une largeur ℓ déterminées, nous sommes sûrs de rencontrer la forme en examinant, par exemple, tous les points situés sur une spirale.

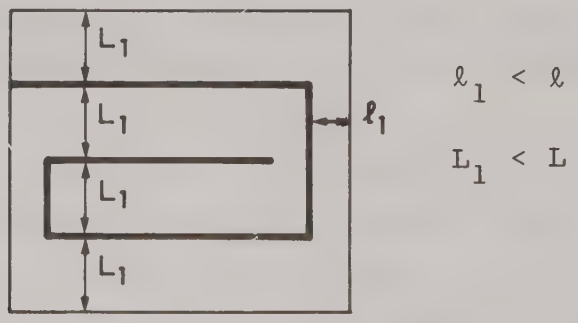


Figure 16 _ Stratégie de recherche d'une forme de dimension ℓ x L.

En chaque point de l'image situé sur ce chemin, l'algorithme calcule la valeur du gradient.

Dès que l'on détecte un maximum local d'amplitude supérieur à un seuil appelé seuil de poursuite, on fait appel à l'algorithme de suivi de contour en lui transmettant :

- le point à gradient localement maximal,
- le gradient lui-même (direction et amplitude)

III.3.2. Algorithme de poursuite

L'algorithme de poursuite cherche à effectuer un parcours à coût minimal dans le graphe limité à quatre générations.

Le sommet du graphe est, à l'origine, le point trouvé par III.3.1. Partant de ce point, $3^4 = 81$ trajectoires sont possibles sur ce graphe. La méthode rigoureuse consiste à calculer le coût correspondant à chaque trajectoire afin de choisir celui à coût minimal.

En théorie, un algorithme basé sur une telle méthode doit parvenir à traverser une zone de perturbations plus aisément, car la poursuite se prolonge de toute manière (même si le gradient devient très faible). Cependant, afin de garder un contour significatif, une interprétation doit être faite à la fin de ce parcours pour décider si celui-ci est valable ou non (c'est à dire est-ce que les quatre derniers points trouvés peuvent être inclus dans la ligne de contour).

On éliminera alors la chaîne si le coût nécessaire à son parcours est trop important.

A ce stade, la feuille correspondant à l'extrêmité de la ligne de contour retenue deviendra elle-même sommet pour une nouvelle chaîne, et l'algorithme recommence.

III.3.3. Heuristiques : parcours du graphe

Un examen systématique des 81 possibilités de parcours prendrait un temps beaucoup trop important ; d'autre part, un examen attentif du graphe ne se révèle nécessaire que rarement dans l'image.

L'idée de cette heuristique consiste à chercher un parcours à coût "satisfaisant" : ce qui ne correspond pas toujours au parcours optimal mais présente l'avantage d'être beaucoup plus rapide à l'exécution.

Pratiquement, l'algorithme agit comme suit :

A chaque génération, choisir la branche à coût minimal, maximiser ce coût en lui substituant une très grande valeur. Enregistrer (empiler) toutes les valeurs calculées pour le père et considérer le fils comme générateur. Calculer alors les trois coûts correspondant à la nouvelle génération. Si aucun de ces coûts n'est satisfaisant (supérieur à un seuil dit "coût maximal permis"), alors revenir au père (dépiler) et prendre la nouvelle branche à coût minimal. Si toutes les branches ont été explorées, alors revenir au grand-père (dépiler à nouveau) et ainsi de suite (au besoin jusqu'au sommet).

Par une telle méthode, on peut explorer la totalité du graphe si cela s'avère nécessaire.

Ainsi, si la poursuite a été déroutée par suite de la traversée d'une zone bruitée, cela se traduira sur l'algorithme par l'augmentation brutale du coût dans la poursuite. Si ce coût devient supérieur au coût maximal permis, l'algorithme renoncera à poursuivre cette ligne et recherchera une nouvelle voie en "remontant" dans le graphe.

Dès qu'un parcours se révèle satisfaisant, on réinitialise la pile et l'algorithme est prêt à la recherche d'un nouveau parcours.

Nous remarquons expérimentalement que, dans la grande majorité des cas, le parcours du graphe s'effectue en

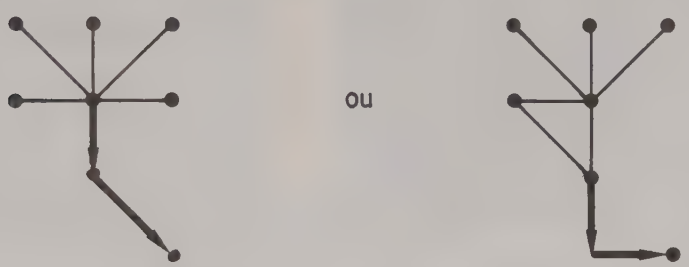
une seule tentative. Seule la traversée de zones fortement perturbées fait appel à cette "remontée" dans le graphe qui consomme un peu de temps de calcul.

III.3.4. Abandon de la poursuite

L'algorithme abandonne la poursuite si au cours du parcours :

- il rencontre un point déjà traité,
- il rencontre le bord de l'image,
- à la fin du parcours, le gradient a une amplitude trop faible ou si on trouve un écart trop important entre la direction du gradient et celle de son père.

On ne s'autorise que les écarts maximaux de type :



L'abandon de la poursuite se traduit par un retour à l'étape III.3.1.

A la fin de ce traitement, on dispose de chaînes de Freeman correspondant aux contours détectés. Les points de départ de ces chaînes sont aussi mémorisés.

III.4. CAS SINGULIERS OU AMBIGUS

Certaines singularités peuvent apparaître lors de la poursuite, c'est le cas, par exemple, lorsque le contour se divise.

Notre structure arborescente peut, en théorie, traiter les aiguillages (qui se traduisent par deux possibilités de parcours à faible coût).

Cependant, la difficulté d'interprétation d'un aiguillage nous a conduits à abandonner provisoirement l'étude de ces cas et à prendre par convention le chemin à coût minimal pour chaque possibilité de succession (voir III.3.3.).

En effet, dans les cas concrets de traitement, il n'est pas possible de prendre arbitrairement une décision sans apport d'informations contextuelles : prenons, pour s'en convaincre, deux cas concrets d'aiguillages :

* Reflet sur un objet (voir la figure 12)
* Objet avec ombre (figure 17f).

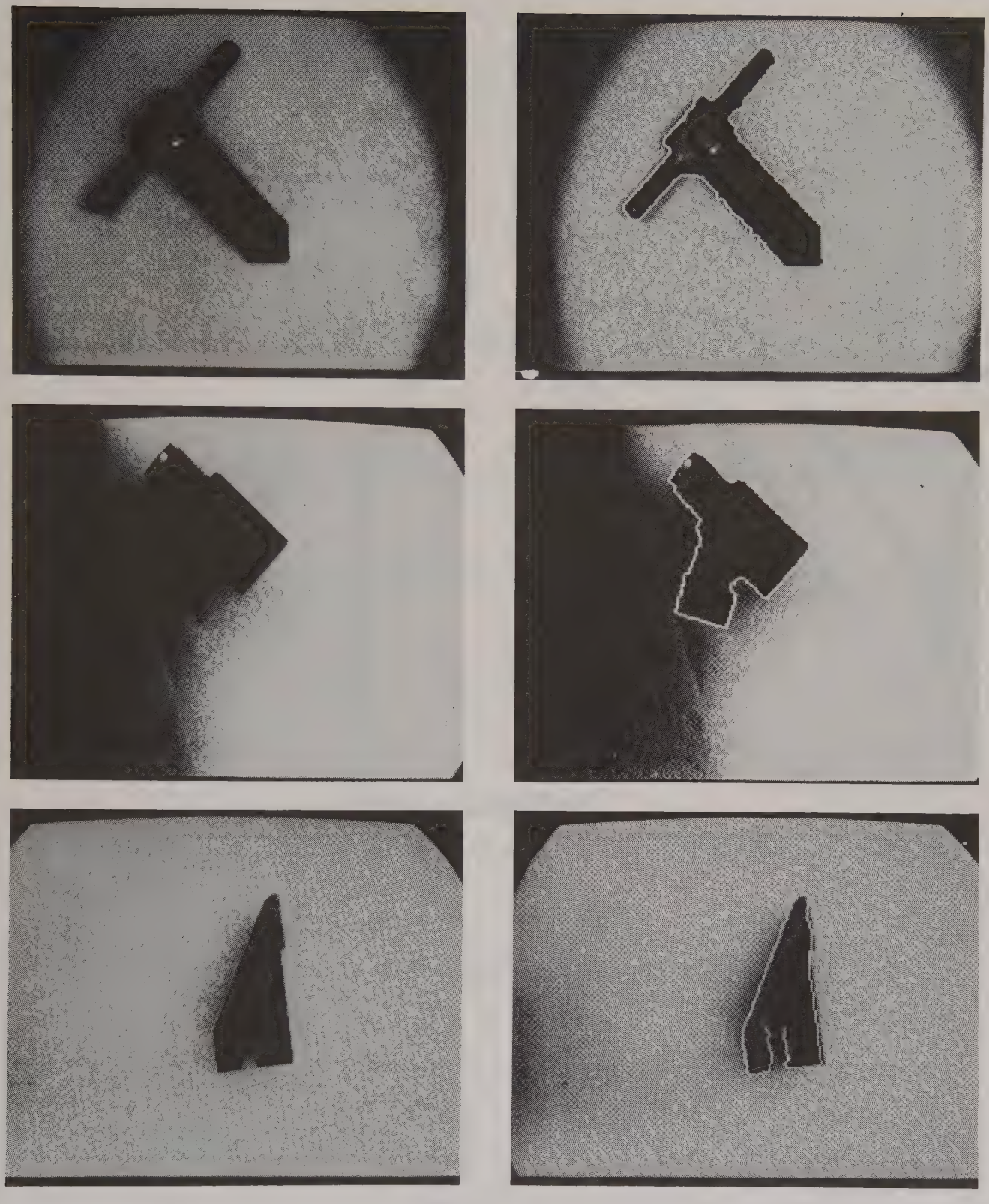
L'obtention du contour propre à l'objet fait intervenir des notions totalement contradictoires;le cas du reflet, il faut choisir l'option maximalisant le contour, alors que pour l'ombre, l'option inverse doit être choisie.

Une telle décision doit faire intervenir des notions d'intelligence artificielle qui compliqueraient "singulièrement l'algorithme.

III.5. RESULTATS

Nous avons testé l'algorithme dans des cas d'images contrastées à peu contrastées. Les résultats ont été satisfaisants.

On remarquera les problèmes dus aux ombres sur l'objet (figure 17e et f), ce qui a amené l'algorithme à abandonner sa poursuite (le coût devenant excessif car le contraste est insuffisant).



A	B
C	D
E	F

FIGURE 17 : Résultats

a) Image originale b) contours suivis (t=0,15 + 0,42s)

c) Image originale d) contours suivis (t=0,15 +0,3s)

e) Image originale f) contours suivis (ABANDON)
(t=0,15 +0,28s)

FIGURE 18 : Représentation codée selon Freeman

Codes :

> 0 →

/ 1 ↗

∧ 2 ↑

\ 3 ↖

< 4 ←

/ 5 ↙

∨ 6 ↓

\ 7 ↘

PARTIE B

RECONNAISSANCE DES IMAGES

"Tout se passe comme si, dans cet ensemble d'images que j'appelle l'univers rien ne pouvait se produire de réellement nouveau que par l'intermédiaire de certaines images particulières dont le type m'est donné par mon corps".

Henri BERGSON

CHAPITRE IV : SEGMENTATION-COMPRESSION

IV.1. INTRODUCTION

Dès que l'on a affaire à un problème de reconnaissance, il est primordial de chercher à comprimer au maximum le nombre de données à analyser.

Nous avons vu avec l'extraction une première étape assez conséquente dans la compression de l'information (pour l'image représentée Figure 18, de 256 x 256 x 4 bits, nous sommes passés à une chaîne de 600 éléments codés sur 3 bits d'où un rapport de compression de :

$$\frac{256 \times 256 \times 4}{600 \times 3} = 145)$$

On peut, néanmoins, poursuivre cette compression en utilisant des techniques d'approximation de contour. On trouvera décrit dans les paragraphes qui suivent :

- Une technique de compression par approximation linéaire des lignes de contour.
- Une technique de codage invariante des segments.
- Une compression conséquente mais avec perte importante de l'information : la technique de repérage des gros segments.

IV.2. SEGMENTATION LINEAIRE DU CONTOUR

IV.2.1. Principe de l'algorithme

Nous désirons approcher le contour par une succession de segments de droite en tolérant un écart maximimum ε autour de la droite.

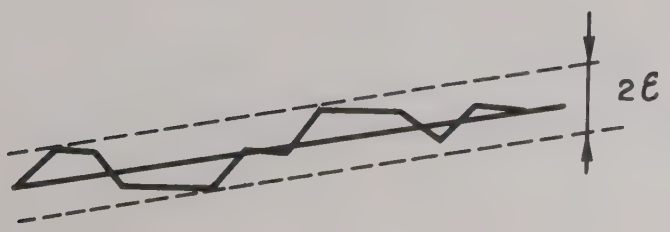


Figure 19 : approximation rectiligne de la ligne de contour.

Plusieurs auteurs ont traité de ce problème dans divers repères. Freeman propose une méthode nécessitant le retour au repère cartésien avec la méthode de Ramer [20]. Pour notre part, nous avons cherché à utiliser la chaîne de Freeman elle-même afin d'obtenir un traitement plus rapide.

L'idée de base de l'algorithme consiste à dérouler la chaîne de Freeman et à tester la possiblité pour chaque élément de la chaîne d'appartenir à une droite, compte-tenu de l'erreur ε tolérée. Si on trouve un élément de la chaîne incompatible avec la droite, celui-ci est considéré comme point d'arrêt. On stocke alors les coordonnées cartésiennes du point de l'image correspondant à l'élément rencontré et l'on recherche une nouvelle droite à partir de ce point. On a alors créé un segment.

IV.2.2. Addition de vecteurs de Freeman

Chaque élément a_i de la chaîne représente l'angle par rapport à l'horizontale du vecteur élémentaire :

Figure 20 : Codage de Freeman.

$$\theta = a_i \times \frac{\pi}{4}$$

Modulo 2π

L'addition vectorielle de 2 éléments de la chaîne forme un vecteur dont l'angle avec l'horizontale s'obtient en faisant la demi-somme des éléments de la chaîne. L'angle obtenu est alors défini modulo π (à cause de la division par 2). Cette règle d'addition n'est valable que pour des segments formant entre eux des angles compris entre $-\frac{\pi}{2}$ et $+\frac{\pi}{2}$ (condition de lissage) comme le montre le tableau suivant :

Addition codée selon Freeman	Codage angulaire	Représentation	Remarque
0 + 0	$0 + 0 \rightarrow 0$		
0 + 1	$0 + \frac{\pi}{4} \rightarrow \frac{\pi}{8}$		
0 + 2	$0 + \frac{\pi}{2} \rightarrow \frac{\pi}{4}$		
0 + 3	$0 + \frac{3\pi}{4} \rightarrow \frac{\pi}{2}$		Exception
0 + 4	$0 + \pi \rightarrow ?$		Exception
0 + 5	$0 + \frac{5\pi}{4} \rightarrow -\frac{\pi}{2}$		Exception
0 + 6	$0 + \frac{3\pi}{2} \rightarrow -\frac{\pi}{2}$		
0 + 7	$0 + \frac{7\pi}{4} \rightarrow -\frac{\pi}{4}$		
1 + 0	$\frac{\pi}{4} + 0 \rightarrow \frac{\pi}{8}$		
1 + 1	$\frac{\pi}{4} + \frac{\pi}{4} \rightarrow \frac{\pi}{4}$		
1 + 2	$\frac{\pi}{4} + \frac{\pi}{2} \rightarrow \frac{3\pi}{8}$		
1 + 3	$\frac{\pi}{4} + \frac{3\pi}{4} \rightarrow \frac{\pi}{4}$		
1 + 4	$\frac{\pi}{4} + \pi \rightarrow \frac{\pi}{2}$		Exception
1 + 5	$\frac{\pi}{4} - \frac{\pi}{4} \rightarrow ?$		Exception
1 + 6	$\frac{\pi}{4} + \frac{3\pi}{2} \rightarrow 0$		Exception
1 + 7	$\frac{\pi}{4} + \frac{7\pi}{4} \rightarrow 0$		

Pratiquement :

$$\text{si } (|a_i - a_j|)_* \leq 2 \text{ alors } \theta_{res} = (a_i + a_j) \times \frac{\pi}{8} \quad \text{Modulo } \pi$$

On peut prolonger ce raisonnement pour l'addition de 2^n vecteurs. Ainsi, l'angle formé par 4 vecteurs de Freeman s'obtient :

$$\text{si } (|a_i - a_j|)_* \leq 2 \text{ alors } \theta_{res} = \left(\sum_{n=k}^{n+4} a_k \frac{\pi}{16}\right) \quad \text{Modulo } \frac{\pi}{2}$$

$$(i,j)/n \leq i < j \leq n+4$$

* Opération modulo 8.

IV.2.3. Algorithme de segmentation

Définition : On appelle mini-segment le vecteur résultant de l'addition vectorielle de 4 vecteurs successifs de Freeman. Supposons que l'on désire segmenter la chaîne de Freeman à partir d'un point A_i, l'origine du segment sera ce point.

L'algorithme de segmentation consiste à prendre dans la chaîne un élément sur deux et à vérifier pour chaque mini-segment dont l'origine est ce vecteur :

- la condition de lissage pour chaque mini-segment
- que l'orientation du mini-segment ne s'écarte pas d'une valeur supérieure à un seuil donné, de l'orientation de référence du segment.

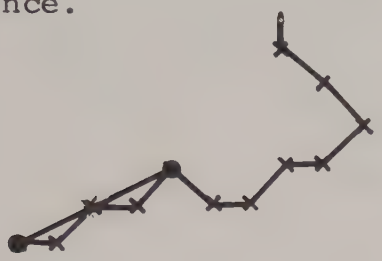
L'orientation de référence du segment est donnée par l'orientation du premier mini-segment considéré (origine).

Si une de ces conditions est vérifiée, le mini-segment est dit mini-segment d'arrêt et l'algorithme va alors examiner l'orientation de chacun des 4 vecteurs de Freeman le constituant pour déterminer un point d'arrêt

correspondant à l'extrêmité du segment A_{i+1}.

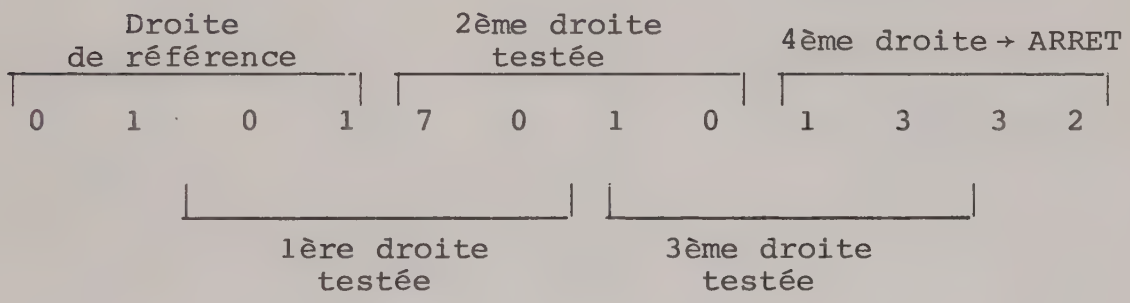
Ce point d'arrêt sera pris pour origine d'un nouveau segment à détecter.

Autrement dit, les 4 premiers vecteurs de la chaîne définissent une droite d'extrapolation qui sert de référence.



chaîne deb. 0,1,0,1,7,0,1,0,
1,3,3,2...

On considère ensuite l'orientation des droites d'extrapolation constituées par 4 vecteurs successifs pris par recouvrement de 2.



Dès qu'une de ces droites forme un angle supérieur à un seuil θ_S avec la référence, on impose un point d'arrêt, après examen minutieux des 4 vecteurs, en un des points de cette droite.

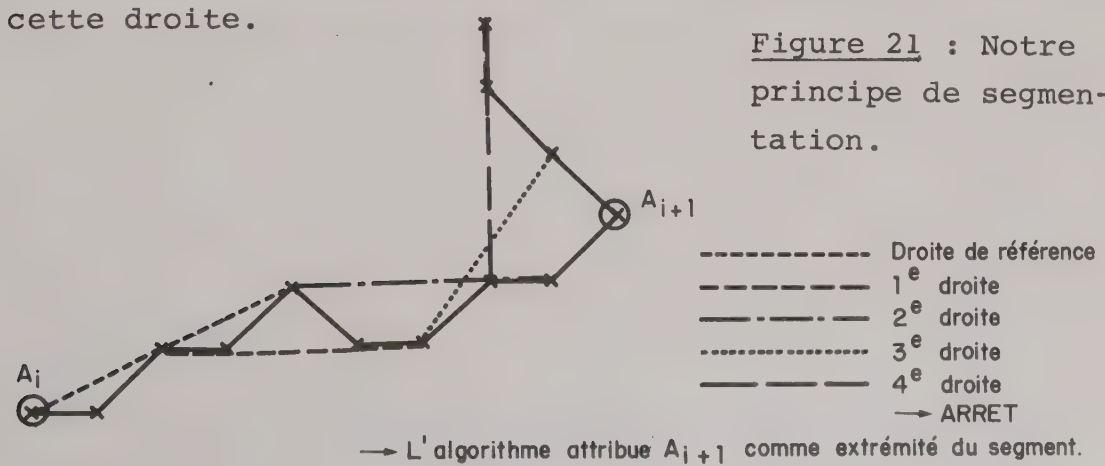
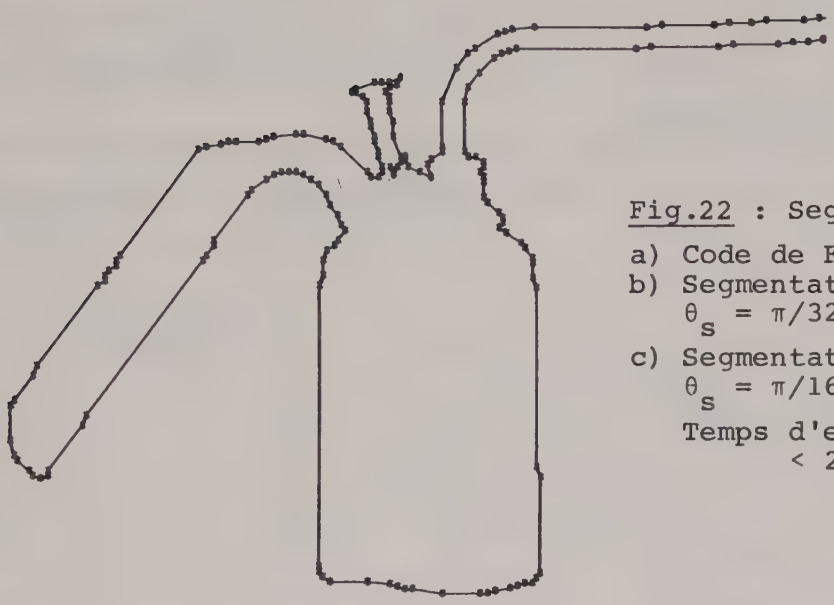


Figure 21 : Notre principe de segmentation.

Une telle méthode est moins précise que le test de Ramer mais présente l'avantage d'être très rapide. Pratiquement, les résultats obtenus sont très satisfaisants, comme le montre la figure 22.



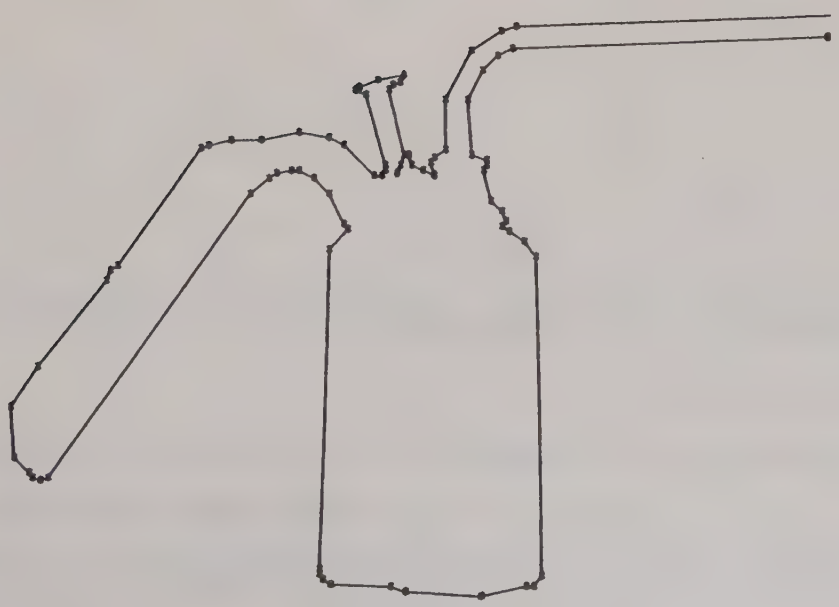


Fig.22 : Segmentation

a) Code de Freeman
b) Segmentation avec $\theta_s = \pi/32$
c) Segmentation avec $\theta_s = \pi/16$

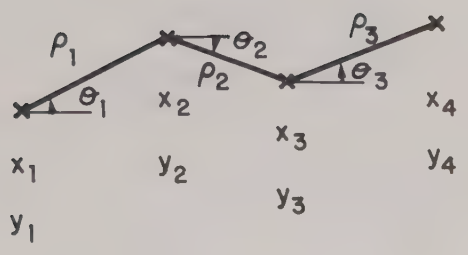
Temps d'exécution < 20 mS

IV.3. CODAGE INVARIANT

A l'issue de l'étape IV.2., nous disposons des coordonnées cartésiennes des points correspondant aux extrêmités des segments. Ces points sont stockés dans l'ordre correspondant au sens du parcours sur la ligne de contour.

Compte tenu de notre objectif (I.3.), nous désirons reconnaître une forme, quelle que soit sa position et son orientation sur l'image. Il est donc intéressant de coder ces segments d'une manière invariante tant en translation qu'en rotation.

Le codage polaire curviligne répond presque parfaitement à ce double objectif : il consiste à enregistrer l'abscisse curviligne de chaque sommet de segment, ainsi que l'orientation correspondante.



Coordonnées cartésiennes			Coordonnées polaires relatives
x_1	y_1	(point de départ)	0 , θ_1
x_2	y_2		ρ_1, θ_2
x_3	y_3		ρ_2, θ_3
x_4	y_4		ρ_3, -

$$\rho = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

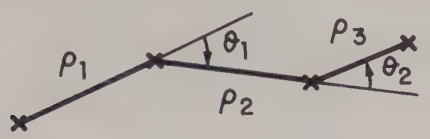
$$\theta = \text{Arc tg} \left(\frac{y_2 - y_1}{x_2 - x_1} \right)$$

Figure 23 : Le codage polaire curviligne

Pour un teI codage,

- une translation de l'ensemble de la figure est invariante,
- une rotation se traduit par une "composante continue" sur l'ensemble du codage : chaque composante se voit augmentée de la valeur de l'angle de rotation.

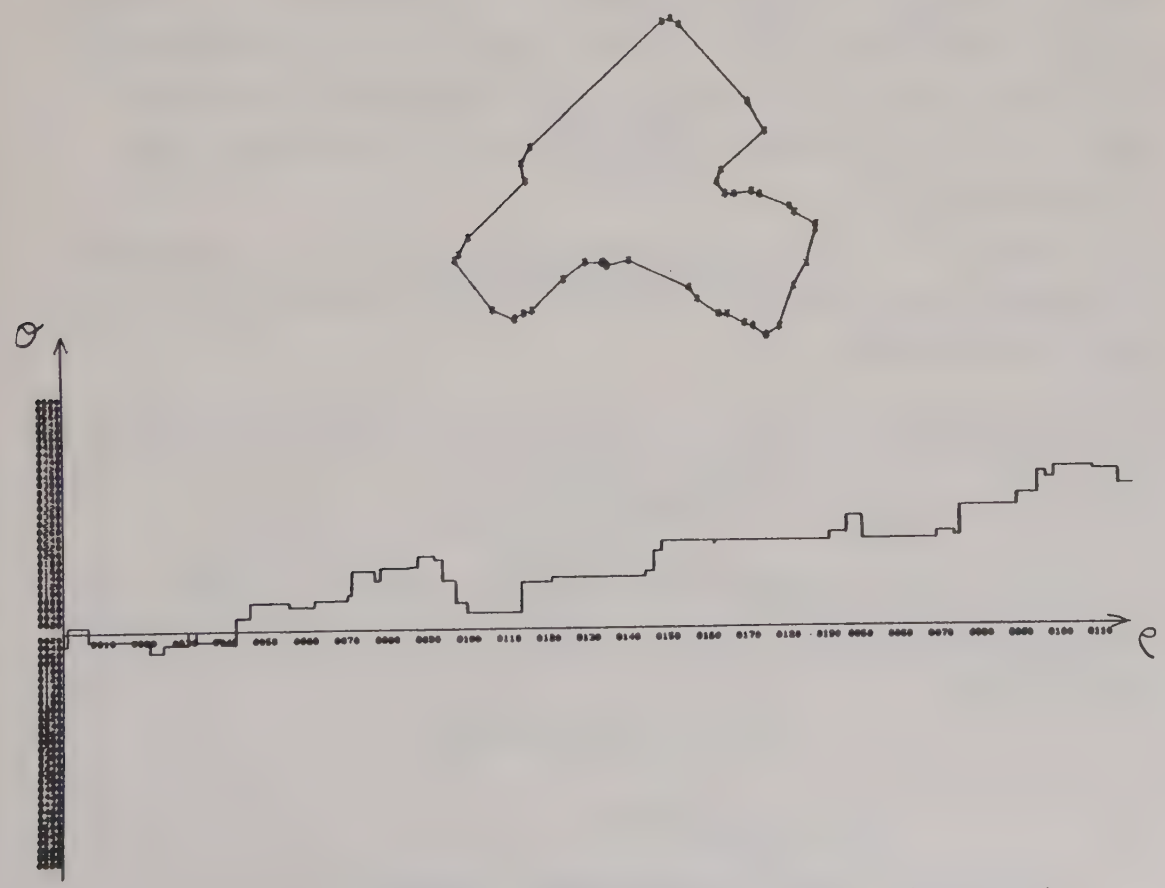
Remarque : Un codage invariant en rotation s'obtient en codant les angles relatifs entre chaque segment :



Cependant une telle représentation n'est pas souhaitable sur le plan du calcul, car elle amène un report incrémental de l'erreur (on ne peut connaître l'angle absolu d'un segment qu'après examen de tous les segmens précédents).

On appelle <u>codage développé</u> la courbe obtenue en portant sur un repère cartésien :

- en abscisse, l'abscisse curviligne des segments,
- en ordonnée, l'angle formé entre le segment considéré et l'horizontale.



<u>Figure 24</u> : Exemple de codage développé

Le codage développé d'un cercle est une rampe ; celui d'un polygone régulier est un escalier à marches régulières.

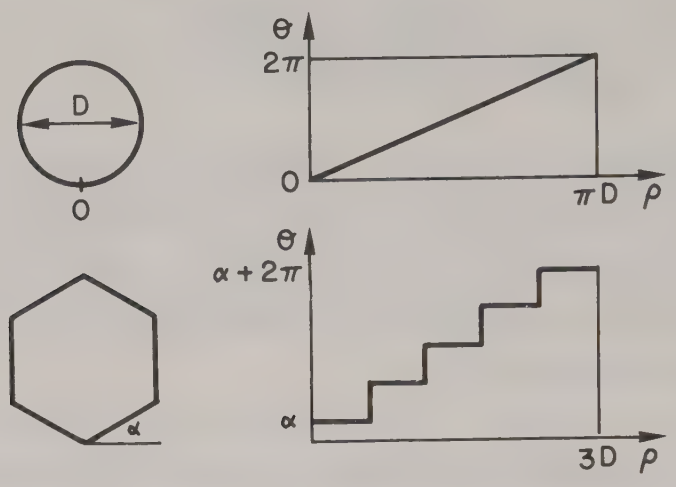


Figure 25 : Codage de quelques formes simples

Propriété : tout contour fermé aura un décalage de ± 2π entre l'origine et l'extrêmité de son codage développé.

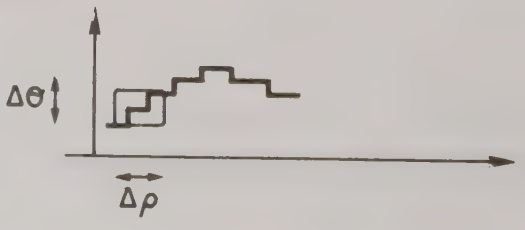
IV.4. COMPRESSION : EXTRACTION DES GROS SEGMENTS

Jusqu'à présent, toutes les données inhérentes au contour ont été conservées. Nous sommes capables de reconstruire la forme à partir du tableau (ρ,θ) précédent quasiment sans perte d'informations (sinon celle occasionnée par l'extrapolation).

A ce stade, il n'est plus possible de comprimer simplement ces données sans amener une dégradation conséquente lors de la restitution.

Cette étape peut s'appeler plutôt "extraction de caractéristiques" car à l'issue du traitement, il n'est plus possible de restituer le contour initial.

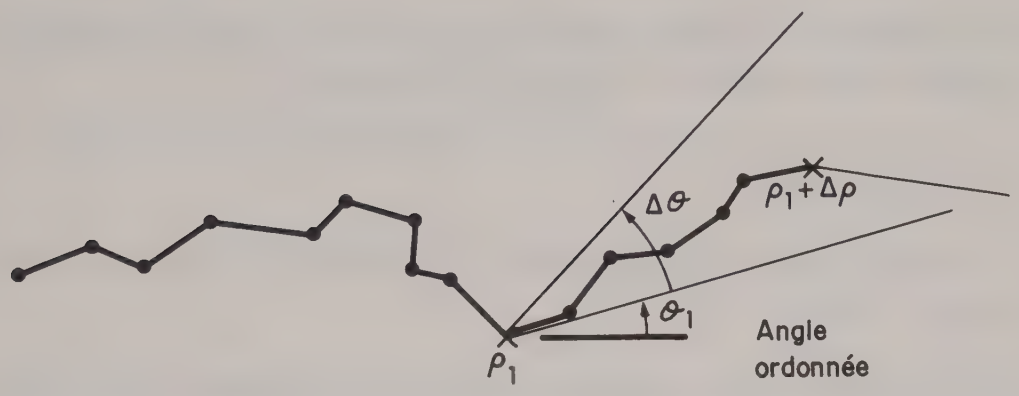
On appelle gros segment toute succession de segments dont le codage développé s'inscrit dans un rectangle de dimension (Δρ,Δθ).



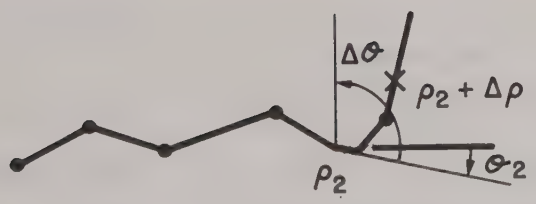
Si $\Delta\rho \gg \Delta\theta$, le gros segment est dit caractéristique rectiligne de longueur $\Delta\rho$.

Si $\Delta\rho \ll \Delta\theta$, le gros segment est dit caractéristique d'angle $\Delta\theta$.

Pratiquement, une caractéristique signifie sur le contour que la ligne de la longueur $\Delta\rho$ est incluse dans un triangle d'angle $\Delta\theta$ et que tous les segments sur cette ligne ont une direction comprise entre θ_1 et $\theta_1 + \Delta\theta$.



Caractéristique rectiligne



Caractéristique d'angle.

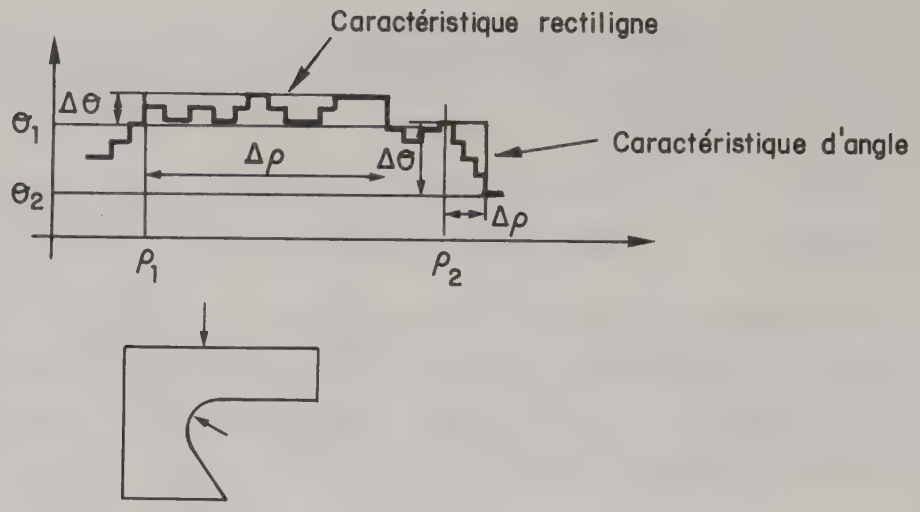


Figure 27 : Gros segments caractéristiques

L'algorithme d'extraction de caractéristiques consiste à calculer pour chaque origine de segment :

- la plus grande longueur $\Delta\rho$ compatible avec un écart d'angle imposé $\Delta\theta_o$,
- le plus grand écart d'angle $\Delta\theta$ compatible avec la longueur imposée $\Delta\rho_o$.

L'algorithme classe ensuite les gros segments caractéristiques dans deux tableaux avec leurs coodonnées curvilignes :

- caractéristiques rectilignes : $\Delta\rho_i$, ρ_i, θ_i par ordre de $\Delta\rho_i$ décroissants,
- caractéristiques d'angle : $\Delta\theta_j$, ρ_j, θ_j par ordre de $\Delta\theta_j$ décroissants.

Le classement s'arrête dès que la caractéristique devient non significative (par exemple dès qu'elle tombe en dessous de la plus grande caractéristique divisée par 3).

Exemple : pour le raccord de la figure 28, nous avons obtenu 3 caractéristiques rectilignes pour $\Delta\theta_o = 40°$, 4 caractéristiques d'angles pour $\Delta\rho_o = 15$ pixels, ces gros segments sont marqués sur le contour initial.

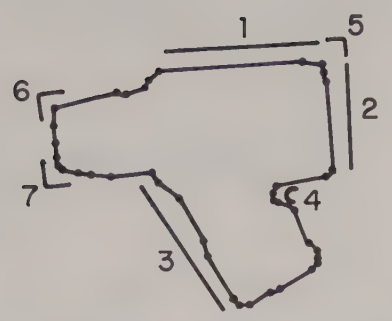


Figure 28 : Gros segments détectés sur le raccord

Remarque : Les gros segments font, en quelque sorte apparaître la dissymétrie circulaire de la forme. Si on applique l'algorithme sur un cercle, on trouve une quantité importante de gros segments qui sont, en outre, non significatifs. Il convient d'éliminer ce cas par un test préalable sur la circularité de l'objet.

CHAPITRE V : RECONNAISSANCE

V.1. DEUX METHODES TRADITIONNELLES

V.1.1. Introduction

Nous avons vu, tout au long de notre exposé, que le traitement d'une image en vue de faire de la reconnaissance se limite souvent à une compression plus ou moins astucieuse des données.

Cependant, si toute compression a pour effet de simplifier et d'accélérer le traitement lors de la reconnaissance, elle s'accompagne toujours d'une perte d'informations sur la forme à analyser.

Nous allons, dans ce paragraphe, rappeler deux méthodes classiques de reconnaissance utilisant ce schéma :

- une méthode utilisant au maximum la compression d'informations : la méthode des moments globaux,
- une méthode basée sur le contour n'utilisant pas de compression : la méthode des masques.

Des méthodes plus élaborées sont décrites dans la bibliographie : [13], [11], [21], [22].

Nous considèrerons à partir de maintenant que les formes sont connexes, c'est à dire délimitées par un contour fermé.

V.1.2. Méthode des moments invariants

Supposons que la forme à reconnaitre soit connexe et binarisée (c'est à dire tout point dans la forme vaut 1, tout point en dehors vaut 0). Il n'y a pas de trous dans la forme. Soit B(x,y) la fonction binaire définissant la forme dans l'image.

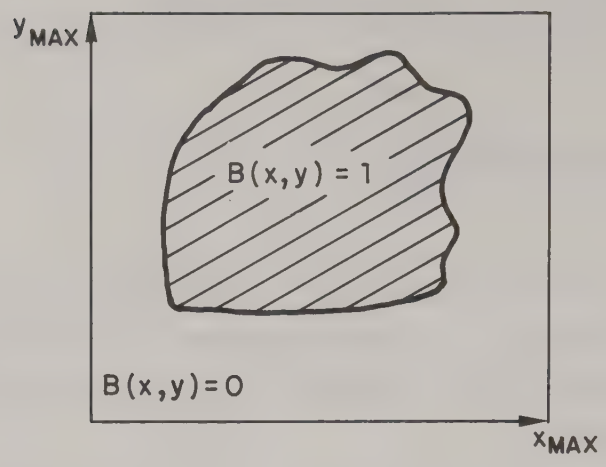


Figure 29 : Définition de B(x,y)

On définit le centre de gravité de la forme par le barycentre de l'image pondérée par la forme binarisée :

$$G_x = \frac{\sum_{x=0}^{xMAX} \sum_{y=0}^{yMAX} x\, B(x,y)}{\sum_{x=0}^{xMAX} \sum_{y=0}^{yMAX} B(x,y)}$$

$$G \begin{pmatrix} G_x \\ G_y \end{pmatrix}$$

$$Gy = \frac{\sum \sum y\, B(x,y)}{\sum \sum B(x,y)}$$

On définit le moment d'inertie de la forme par :

$$M_x = \frac{\sum_{x=0}^{xMAX} \sum_{y=0}^{yMAX} x^2\, B(x,y)}{\sum_{x=0}^{xMAX} \sum_{y=0}^{yMAX} B(x,y)}$$

$$M \begin{pmatrix} M_x \\ M_y \end{pmatrix}$$

$$M_y = \frac{\sum \sum y^2\, B(x,y)}{\sum \sum B(x,y)}$$

On définit par Moment centré tout moment ramené au centre de gravité. Ces moments sont, par construction même, invariants à toute translation de la forme dans l'image. Il existe certains moments invariants par rotation, il s'agit des moments centrés ramenés aux axes d'inertie.

Si on note :

$$m_{ij} = \sum_{x=0}^{xMAX} \sum_{y=0}^{yMAX} x^i \, y^j \, B(x,y)$$

On a :

$$x_G = \frac{m_{10}}{m_{00}} \qquad y_G = \frac{m_{01}}{m_{00}}$$

On exprime alors les moments centrés Y_{ij}

$$Y_{ij} = \sum_{x=0}^{xMAX} \sum_{y=0}^{yMAX} (x - x_G)^i \, (y - y_G)^j \, B(x,y)$$

Les premiers moments centrés sont :

$$\mu_{00} = m_{00}$$

$$\mu_{10} = 0 = \mu_{01}$$

$$\mu_{20} = m_{20} - x_G \, m_{10}$$

$$\mu_{02} = m_{02} - y_G \, m_{01}$$

Si on normalise ces moments centrés :

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(p+q)/2}}$$

On démontre que les moments définis par :

$$C_1 = \eta_{20} - \eta_{02}$$

$$C_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$C_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2$$

$$C_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

sont invariants par translation, rotation et homothétie [27], [28], [29].

La méthode de reconnaissance par moments consiste à enregistrer les valeurs de ces moments pour les différentes formes à reconnaitre, puis à évaluer une distance entre cette description de la forme inconnue et les différentes descriptions des formes connues. Cette distance peut être de la forme Euclidienne :

$$d_{IJ}^2 = \sum_i (C_{iI} - C_{iJ})^2 \times a_i$$

d_{IJ} = distance entre la forme I et la forme J.

a_i = facteur de pondération du moment i.

L'algorithme consiste donc à :

- calculer les moments de la forme inconnue,
- calculer les distances entre la forme inconnue et les formes répertoriées,
- associer la forme inconnue à la forme répertoriée de distance minimale si celle-ci est inférieure à un seuil de rejet défini.

Cette méthode est très simple ; la plupart des systèmes actuels de vision du commerce l'emploient.

Vu le petit nombre de paramètres caractéristiques (typiquement 7 à 15), la reconnaissance est très rapide.

Cependant, plusieurs formes différentes d'aspect peuvent avoir les mêmes valeurs de moments (notamment certains objets et leurs symétriques).

On peut donc contester cette méthode en affirmant que la compression des informations a été trop importante (il est d'ailleurs impossible de recréer la forme par la seule connaissance des valeurs numériques de ces moments).

V.1.3. Méthode des masques

La méthode décrite ci-après n'est pas invariante par rotation. On suppose donc la forme convenablement orientée.

Supposons que la forme soit connexe et la ligne du contour binarisée (1 sur la ligne, 0 ailleurs).

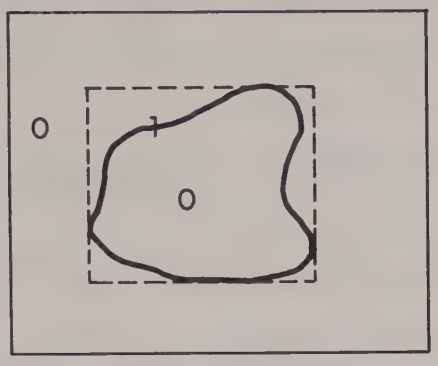


Figure 30 : Définition de B'(x,y)

On appelle boite ou emboitement du contour, le plus petit rectangle constitué par deux segments verticaux et deux horizontaux, contenant la forme.

On appelle contour dilaté de largeur n, la ligne de contour épaissie artificiellement pour constituer une ligne d'épaisseur n donnée.

La méthode des masques consiste à faire coïncider les deux boites des contours à comparer, puis à faire le ET logique entre la première boite et le négatif du contour dilaté de largeur n dans la deuxième boite.

Tout point à 1 dans la boite résultante correspond à une non coïcidence du contour.

Cette méthode est très précise (tolérance = largeur n) mais nécessite une préorientation des formes.

Si cette condition n'est pas réalisée au départ, on doit faire une rotation du contour (opération longue en temps de calcul).

V.2. METHODE PROPOSEE

V.2.1. Sélection par coïncidences des gros segments

Dans la mesure où les gros segments ne sont pas suffisants pour reconstituer le contour, nous pouvons comparer cette méthode à celle des moments invariants.

Nous recherchons, en effet, à faire apparaître des similitudes dans les caractéristiques globales que nous avons extraites : les gros segments.

Deux gros segments sont dits compatibles à (ρ_i, θ_i) prés si :

- leurs longueurs sont égales à ρ_i près
- leurs largeurs sont égales à θ_i près.

Un gros segment A $(\rho_A, \theta_A, \Delta\rho_A, \Delta\theta_A)$ est dit en coïncidence avec un gros segment B $(\rho_B, \theta_B, \Delta\rho_B, \Delta\theta_B)$ pour un décalage (ρ_o, θ_o) si en faisant coïncider l'origine du repère développé définissant A avec le point (ρ_o, θ_o) dans le repère développé définissant B, on vérifie les propriétés suivantes :

- les abscisses curvilignes des gros segments A et B présentent entre elles un écart inférieur à un seul $\Delta\rho_S$ donné :

$$\rho_A - \rho_B - \rho_o < \Delta\rho_s$$

- les angles correspondants aux ordonnées des gros segments A et B présentent entre eux un écart inférieur à un seuil $\Delta\theta_S$ donné :

$$\theta_A - \theta_B - \theta_s < \Delta\theta_s$$

- les dimensions des gros segments sont compatibles à (ρ_i, θ_i) près.

Pratiquement, la coïncidence des deux gros segments caractéristiques signifie que l'on trouve sur les deux formes correspondantes deux lignes caractéristiques "se ressemblant", situées à la même abscisse curviligne et dans

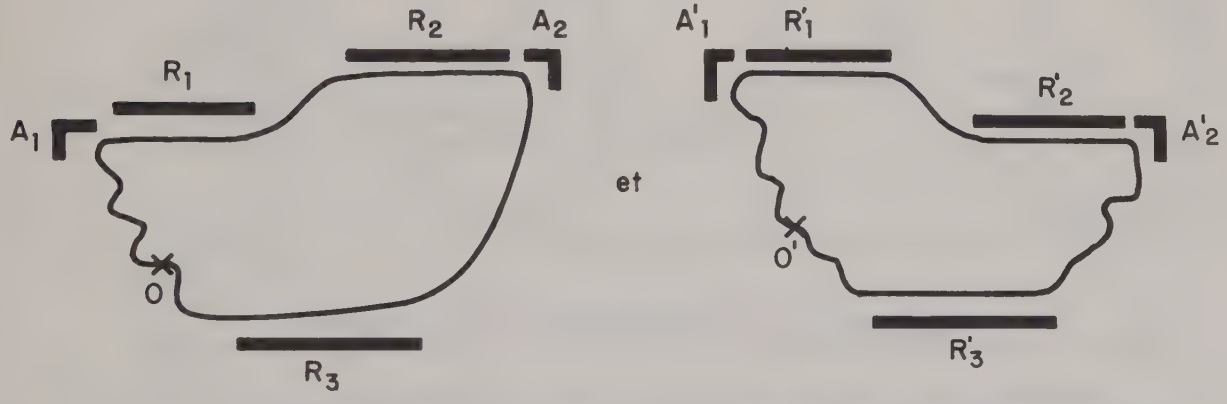
une orientation identique (aux écarts de tolérance près).

L'algorithme de reconnaissance basé sur le principe des coïncidences consistera à essayer de faire coïncider un gros segment de l'objet inconnu avec un gros segment d'un objet de référence. S 'ils satisfont aux critère de ressemblance, alors on calcule le vecteur de décalage (ρ_o, θ_o), puis on recherche si les autres gros segments de l'objet inconnu coïncident avec les gros segments de l'objet de référence.

Le nombre de coïncidences repérées consitue le critère de reconnaissance et on associe l'objet inconnu à la classe d'objets correspondant à la plus forte valeur du nombre de coïncidences.

Dans la pratique, cet algorithme est très rapide car le nombre de gros segments est, en général, limité mais il manque de précision.

Il est possible de trouver deux formes de contours différents ayant les mêmes caractéristiques globales. Par exemple, les deux formes suivantes seront identiques du point de vue de l'algorithme :



Pour ces deux formes, les gros segments repérés coïncident tous si on prend un vecteur de décalage (0, 0).

Nous allons voir (§.V.3.) qu'effectivement cette méthode présente quelques lacunes et que s'il est fort probable que l'objet à reconnaître présente un nombre de coïncidences important avec l'objet de référence dont il est issu, il se peut néanmoins qu'un grand nombre de coïncidences apparaissent aussi avec une autre classe d'objets dont le contour est différent.

Une telle propriété rend délicate la notion de rejet.

Cette méthode de "tri par caractéristiques" s'écarte un peu trop de la notion de contour. Autrement dit, nous avons perdu trop d'informations en ne retenant que ces caractéristiques et l'on peut mettre cette méthode en défaut si l'on choisit judicieusement les objets à reconnaître.

V.2.2. Corrélation sur la représentation invariante

V.2.2.1. Distance d'aire entre deux formes

La première idée qui vient à l'esprit lorsque l'on désire comparer deux formes consiste à essayer de les superposer.

Supposons, en effet que l'on dispose de deux formes $\mathcal{A}$ et $\mathcal{B}$ dessinées sur du papier calque. Soit $\mathcal{R}_A(O_A\ xy)$ le repère solidaire de la forme $\mathcal{A}$ et $\mathcal{R}_B(O_B\ x'y')$, celui solidaire de la forme $\mathcal{B}$. Ces deux repères sont définis dans un repère absolu.

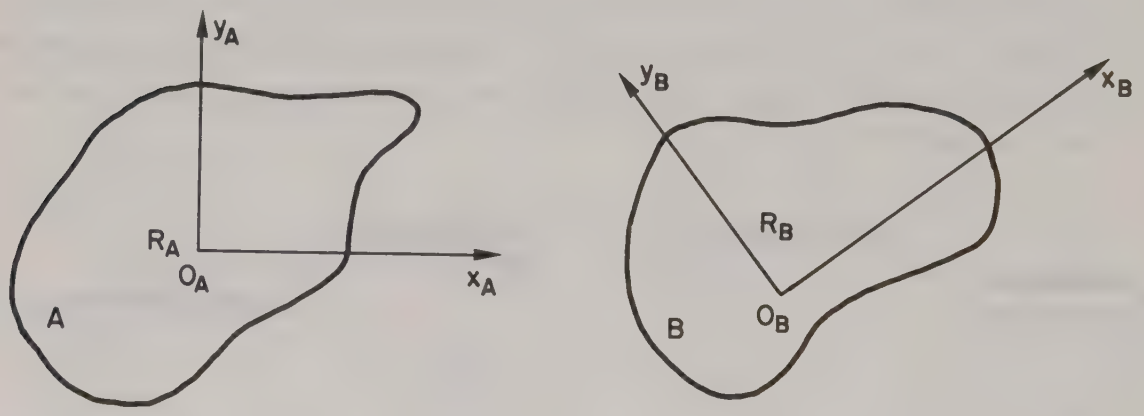


Figure 32 : Comparaison de 2 formes

Soit : $S_{\mathcal{A}}$ la surface délimitée par la forme $\mathcal{A}$.

$S_{\mathcal{B}}$ la surface délimitée par la forme $\mathcal{B}$.

On définit la distance d'aire entre deux formes $\mathcal{A}$ et $\mathcal{B}$ relativement à $\mathcal{R}_A$, $\mathcal{R}_B$ par la surface constituée par l'union exclusive de deux surfaces $S_{\mathcal{A}}$ et $S_{\mathcal{B}}$

$$d(\mathcal{A},\mathcal{B})_{\mathcal{R}_A\,\mathcal{R}_B} = S_{\mathcal{A}} \oplus S_{\mathcal{B}} = (S_{\mathcal{A}} \cup S_{\mathcal{B}}) \cap \overline{(S_{\mathcal{A}} \cap S_{\mathcal{B}})}$$

Dans notre cas précédent, l'intersection entre $\mathcal{A}$ et $\mathcal{B}$ est nulle : la distance d'aire est donc égale à la somme des deux surfaces. Si, maintenant, nous déplaçons le repère $\mathcal{R}_{\mathcal{B}}$ afin que cette intersection existe, la distance d'aire se voit décroitre car diminuée de l'aire de cette intersection.

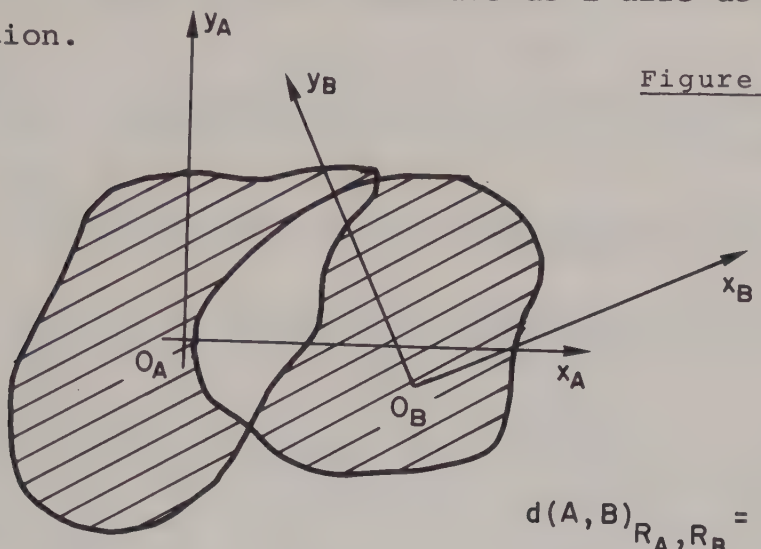


Figure 33: Distance d'aire entre formes

$$d(A,B)_{R_A,R_B} = S_A + S_B - S_{A\cap B}$$

On vérifie que la distance d'aire vérifie bien les trois conditions :

. $d(\mathcal{A},\mathcal{B}) = 0 \Leftrightarrow S_{\mathcal{A}} = S_{\mathcal{B}} \Leftrightarrow \mathcal{A} = \mathcal{B}$

. $d(\mathcal{A},\mathcal{B}) = d(\mathcal{B},\mathcal{A})$

. $d(\mathcal{A},\mathcal{B}) + d(\mathcal{B},\mathcal{C}) \leq d(\mathcal{A},\mathcal{C})$

La distance d'aire est donc une distance au sens Euclidien.

Comparer deux formes $\mathcal{A}$ et $\mathcal{B}$ consiste à chercher la disposition des repères $\mathcal{R}_{\mathcal{A}}$ et $\mathcal{R}_{\mathcal{B}}$ qui minimise la distance d'aire entre les deux formes.

Remarquons que cette notion de distance d'aire et très liée à la réalité : nous réalisons effectivement cette opération de superposition de dessins lorsque l'on désire les comparer.

Pour comparer deux formes, il faut donc calculer la distance d'aire en faisant varier trois paramètres : en effet, si on ramène le repère $\mathcal{R}_{\mathcal{B}}$ relativement à $\mathcal{R}_{\mathcal{A}}$, celui-ci peut s'exprimer dans $\mathcal{R}_{\mathcal{A}}$ par trois composantes :

- les coordonnées de B dans $\mathcal{R}_A$ (B_x, B_y)
- l'angle formé entre $\mathcal{R}_A$ et $\mathcal{R}_B$ $(\widehat{Ax_A, Bx_B})$

Une telle opération est absolument impensable sur le plan du calcul à cause du temps démesuré qu'il faudrait pour l'effectuer.

Nous allons donc chercher des heuristiques afin de limiter le nombre de paramètres à faire varier.

V.2.2.2. Distance entre deux codages développés

Revenons au codage développé de la forme (IV.2). Ce codage constitue une représentation fonctionnelle : il est, en quelque sorte, la signature de la forme. On définit donc la signature d'un contour fermé segmenté par son codage développé dont l'abscisse curviligne est limitée entre 0 et le périmètre du contour. Le périmètre est appelé longueur de la signature.

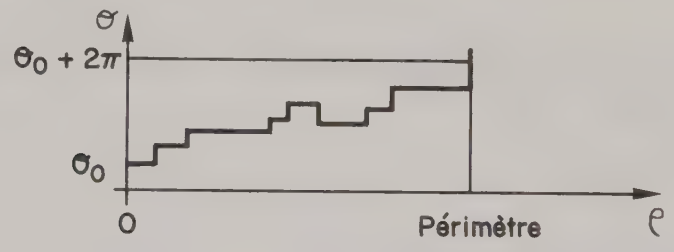


Figure 34 : Signature d'un contour fermé

Nous avons vu que cette représentation est bi-univoque par rapport à celle du contour segmenté dans le domaine de l'image.

Nous allons, dans ce paragraphe, définir une distance entre deux signatures dans le domaine développé, puis examiner la signification de cette distance dans le domaine de l'image.

Tout d'abord quelques définitions :

- On appelle signature normalisée par rapport au repère développé $\mathbb{R}$ la signature ayant une valeur moyenne d'angle nulle dans ce repère.

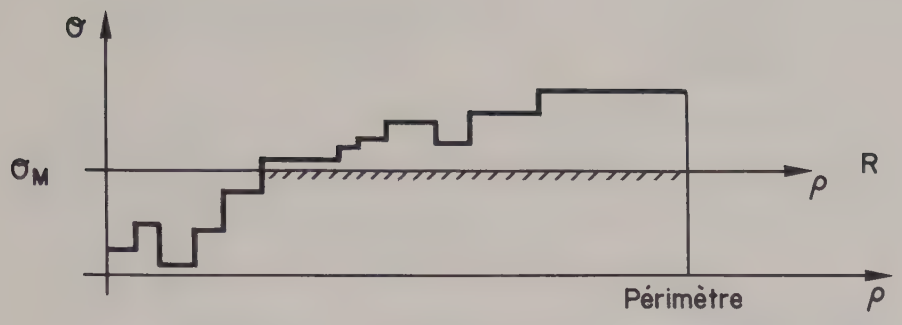


Figure 35 : Signature normalisée d'un contour fermé

On normalise une signature en calculant la valeur moyenne θ_M de son angle sur la totalité de sa longueur, puis en considérant le nouveau repère R translaté de θ_M.

- On appelle signature prolongée le codage développé que l'on obtiendrait en tournant indéfiniment autour de la forme sur son contour dans le domaine de l'image.

  La signature prolongée s'obtient en reportant à l'extrémité de la signature, l'origine de celle-ci translatée de $\pm 2\pi$, puis en portant à l'extrémité du codage obtenu l'origine de la signature translatée $\pm 4\pi$, et ainsi de suite.

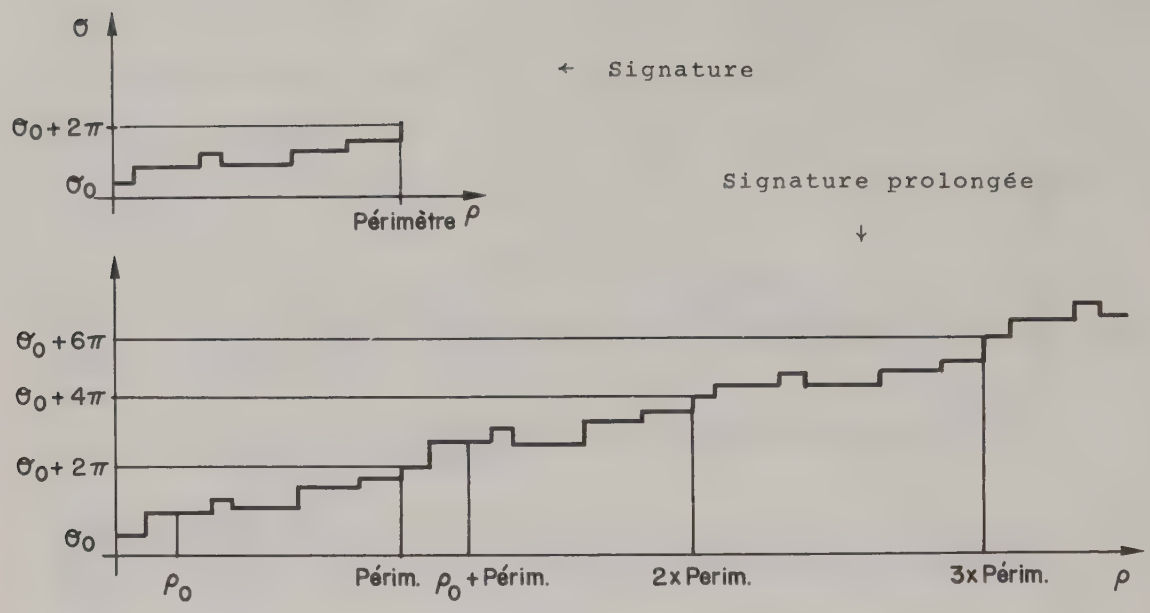


Figure 36 : Signature prolongée d'un contour fermé

- On appelle signature translatée de ρ_o, la portion de signature prolongée comprise entre ρ_o et $(\rho_o + Perim)$.
- On appelle signature dilatée par homothétie de rapport k la signature obtenue en dilatant l'axe des abscisses du rapport k.

Distance entre signatures

On ne peut définir de distance qu'entre deux signatures normalisées et ayant la même longueur.

La distance entre deux signatures 𝔸 et 𝔹 normalisées et de même longueur est égale à la somme en valeur absolue des surfaces délimitées par les segments de 𝔸 et 𝔹 (c'est à dire l'aire entres les deux courbes 𝔸 et 𝔹).

Portons 𝔸 et 𝔹 sur le même repère :

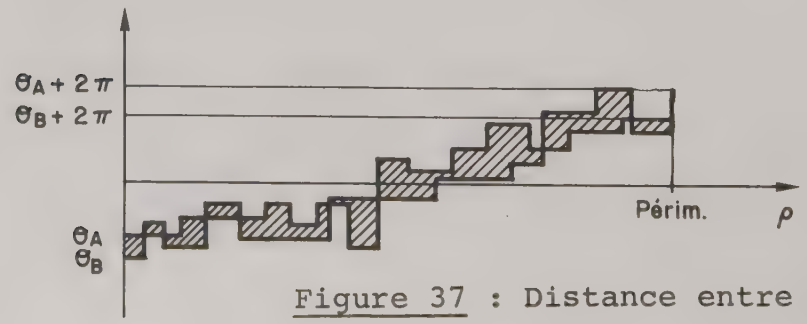


Figure 37 : Distance entre deux signatures

On vérifie que la distance entre deux signatures est Euclidienne (voir Annexe C).

V.2.2.3. Interprétation de la distance dans le domaine de l'image

Nous venons de définir une distance entre deux signatures. Par homomorphisme, nous avons donc défini une distance Euclidienne dans le domaine de l'image, essayons de préciser cette distance.

Tout d'abord, à quelles notions correspondent dans l'image les définitions précédentes ? On vérifiera que :

- une translation en abscisse de ρ équivaut à prendre sur le contour de la forme une origine décalée de ρ,
- une translation en ordonnée de θ équivaut à une rotation globale de la forme d'un angle θ,
- une dilatation par homothétie de rapport k est équivalente à une homothétie sur la forme d'un rapport k,
- la normalisation d'une signature consiste à faire pivoter la forme afin de l'amener dans une orientation arbitraire qui n'a d'ailleurs pas de signification géométrique simple dans l'image.

La distance entre deux signatures de même longueur $\mathbb{A}$ et $\mathbb{B}$ peut aussi s'exprimer dans l'image par une distance entre deux contours de même périmètre :

. Partant en $\mathscr{A}$ d'un point I d'abscisse ρ sur le contour, j'effectue un déplacement élémentaire $d\rho$ sur le contour $\mathscr{A}$, soit $\theta_A(\rho)$ l'orientation de ce segment élémentaire.

. Partant en $\mathscr{B}$ d'un point J situé à la même abscisse ρ sur le contour, j'effectue un déplacement élémentaire $d\rho$ dans le même sens sur le contour $\mathscr{B}$ soit $\theta_B(\rho)$ l'orientation de ce segment élémentaire.

La distance élémentaire entre ces deux segments élémentaires est l'écart en valeur absolue entre les deux directions de ces segments.

$$d(d\,\mathcal{A},\ d\,\mathcal{B}) = |\theta_{A(\rho)} - \theta_{B(\rho)}|\ d\rho$$

Si on continue à parcourir les deux contours, on définit la distance par :

$$d(\mathcal{A},\ \mathcal{B}) = \int_{\rho=0}^{\text{Périmètre}} |\theta_A(\rho) - \theta_B(\rho)|\ d\rho$$

Nous remarquons donc que la distance entre deux signatures exprime, dans l'image, une somme d'écarts de directions entre des segments élémentaires situés aux mêmes abscisses curvilignes.

Pour deux contours fermés quelconques $\mathcal{A}$ et $\mathcal{B}$ de même périmètre, le calcul de cette distance correspond à amener en chaque point de $\mathcal{A}$, d'abscisse curviligne ρ, un segment de droite dont l'orientation est celle relevée en $\mathcal{B}$ à la même abscisse curviligne ρ,

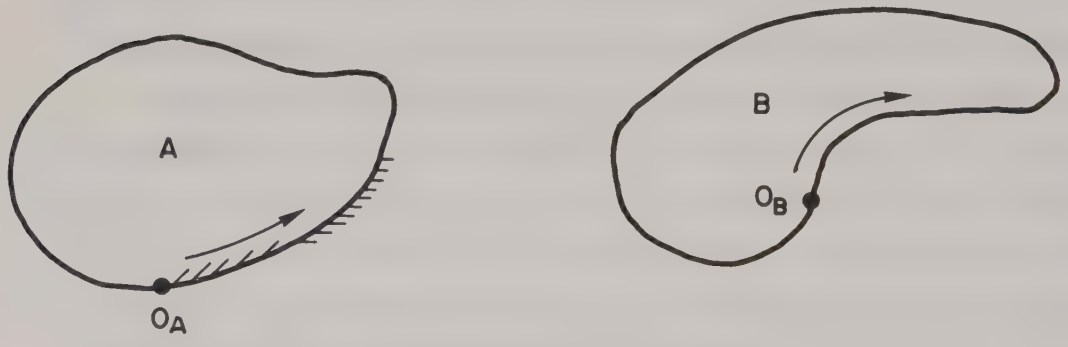


Figure 38 : Signification sur l'image de la distance

et de calculer donc $\int_{\rho=0}^{\text{Périm}_A} |\theta_A(\rho) - \theta_B(\rho)|\,d\rho$.

C'est-à-dire une estimation en chaque point de l'erreur de direction effectuée en traçant le segment élémentaire dans la direction imposée par le contour $\mathcal{B}$ plutôt que dans la direction "de référence" donnée par $\mathcal{A}$.

Remarque : dans le cas d'une représentation segmentée, le calcul de l'intégrale se simplifie car $\theta(\rho)$ reste constant sur toute la longueur d'un segment.

V.2.2.4. "Corrélation" entre deux signatures

Définition :

Nous appellerons corrélation sur une longueur ρ_L, l'opération consistant à rechercher le minimum de distance entre une signature $\mathbb{A}$ normalisée et la signature normalisée B translatée de ρ, pour ρ variant continuement de 0 à ρ_L.

Procédé de détermination :

Dans le doamine de l'image, supposons que :
$\mathcal{A}$ soit un contour d'origine O_A,
$\mathcal{B}$ soit un contour d'origine O_B.

Il s'agit d'appliquer la distance précédente entre $\mathcal{A}$ et $\mathcal{B}$, puis entre $\mathcal{A}$ et $\mathcal{B}$ dont l'origine O'_B serait décalée d'une longueur élémentaire $d\ell$, puis entre $\mathcal{A}$ et $\mathcal{B}$ dont l'origine O''_B serait décalée de 2 $d\ell$, et ainsi de suite jusqu'à ce que l'on arrive en O_B^F d'abscisse curviligne ρ_L par rapport à O_B.

La distance minimale rencontrée au cours de cette succession d'opérations est la valeur de la corrélation entre $\mathbb{A}$ et $\mathbb{B}$ sur la longueur ρ_L.

V.2.2.5. Algorithme de corrélation

Dans la pratique, les signatures réelles ne sont ni normalisées, ni de même longueur. Il faut donc effectuer un prétraitement.

Les formes sont ici supposées fermées et segmentées conformément à IV.2. Supposons que l'on dispose d'une signature normalisée de référence $\mathbb{A}$ et que l'on désire comparer une forme inconnue $\mathcal{B}$, de signature $\mathbb{B}$ à $\mathcal{A}$.

Soit P_A la longueur de $\mathbb{A}$ et P_B la longueur de $\mathbb{B}$.

•La première étape consiste à imposer à $\mathbb{B}$ une homothétie de rapport $\frac{P_A}{P_B}$. Soit $\mathbb{B}'$ la nouvelle signature.

Pratiquement : on dilate le contour $\mathcal{B}$ afin qu'il ait le même périmètre que $\mathcal{A}$.

La deuxième étape consiste à normaliser la signature de $\mathcal{B}$. On calcule donc le vecteur de translation θ_B ramenant la signature à la valeur normalisée.

Pratiquement : on fait pivoter $\mathcal{B}$ d'un angle θ_B.

L'algorithme de corrélation agit ensuite comme suit :

- Calculer la distance entre A et B'.
- Faire $D_M = d(A, B)$
  $T_{HETA} = \theta_B$
  $R_{OECART} = R_{OB} = 0$

CORPS :
- Translater $\mathcal{B}'$ du pas de corrélation $\Delta\rho$
- Faire $R_{OECART} = R_{OECART} + \Delta\rho$
- Normaliser $\mathcal{B}'$ (on obtient un nouveau θ_B peu différent du premier)
- Calculer la distance entre $\mathcal{A}$ et $\mathcal{B}'$
  Si elle est inférieure à D_M, faire :
  $D_M = d(\mathcal{A}, \mathcal{B})$
  $T_{HETA} = \theta_B$
  $R_{OB} = R_{OECART}$
- Si $R_{OECART} < P_A$, aller en CORPS.

A la fin de cet algorithme, nous connaissons la corrélation entre A et B,

+ l'angle T_{HETA} correspond à l'angle optimal entre les deux formes (angle pour lequel la distance est minimale)
+ R_{OB} indique l'écart d'abscisse curviligne entre les deux origines pour lequel la distance est minimale.

Exemple : Pour les deux formes représentant un raccord, dont les codages sont donnés à la figure 40. L'algorithme précédent donne, pour un pas de 1 pixel :

$D_M = 91$
$T_{HETA} = 112,4°$
$R_{OB} = 66$ pixels

La distance minimale correspond à cette superposition.

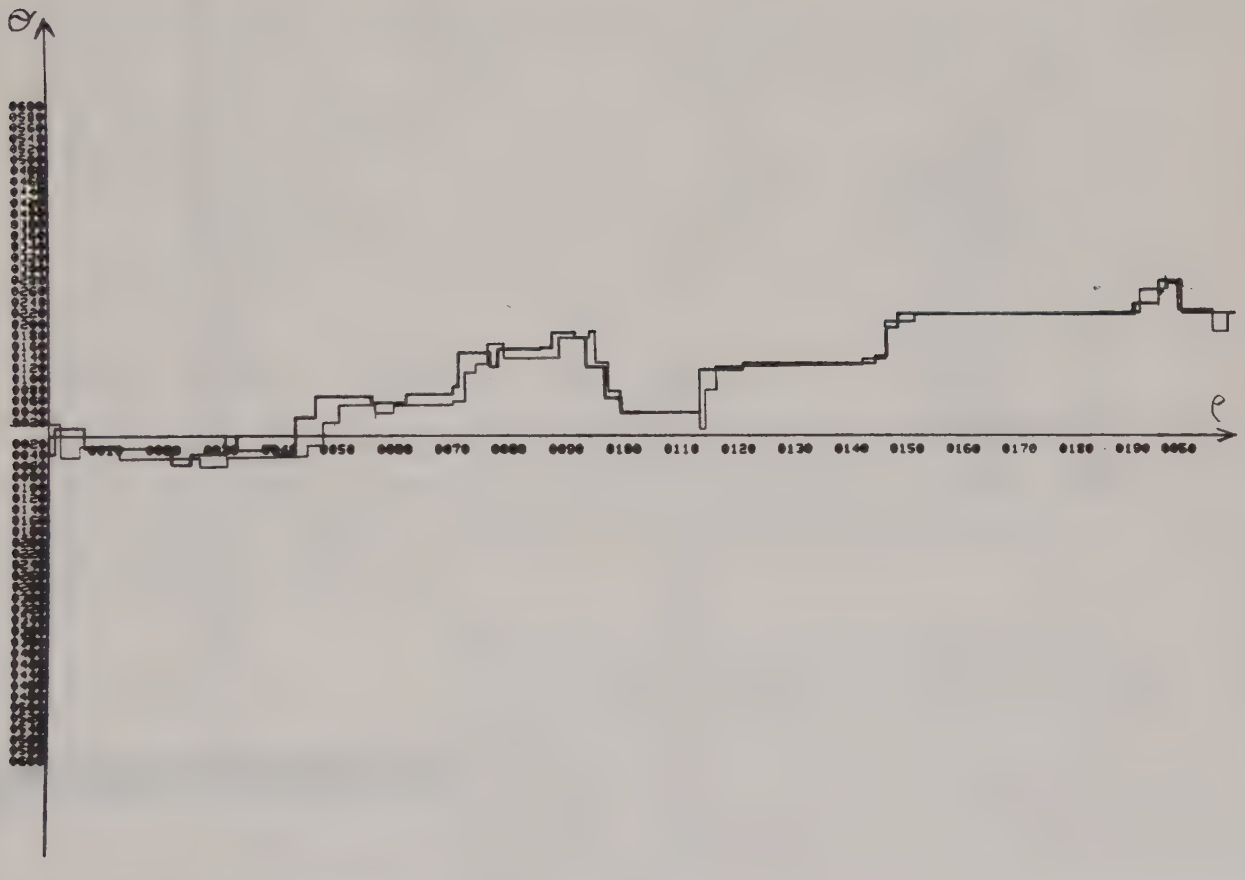
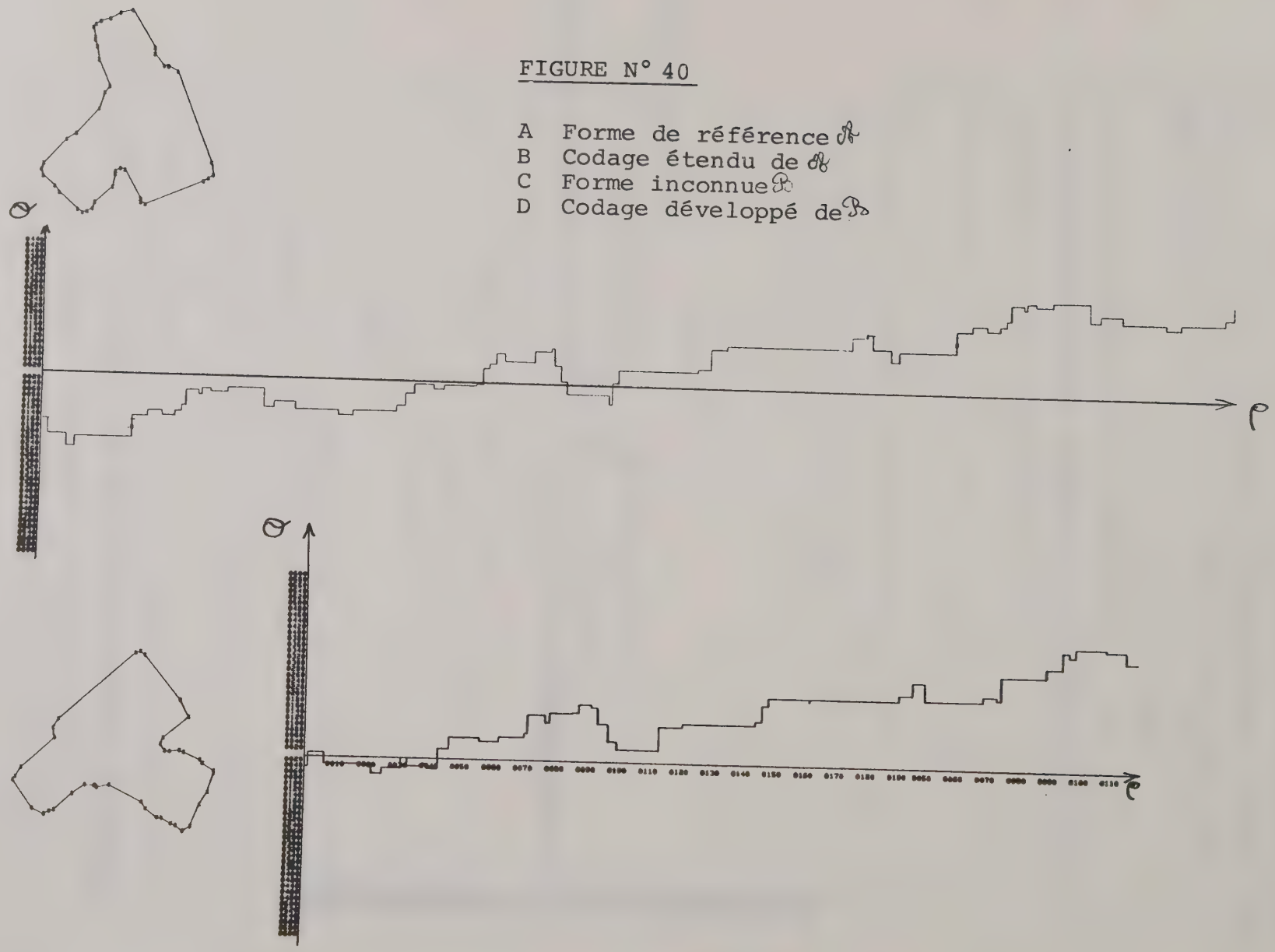


Figure 39 : Corrélation entre les 2 formes de la figure 40

FIGURE N° 40

A Forme de référence $\mathcal{A}$
B Codage étendu de $\mathcal{A}$
C Forme inconnue $\mathcal{B}$
D Codage développé de $\mathcal{B}$



La méthode exposée ci-dessus donne de très bons résultats et fait apparaître la notion d'inspection du contour (bien que la distance n'indique pas le type de déformation). L'inconvénient majeur de cette méthode est sa durée: pour un objet d'environ 500 segments, de périmètre 1000 pixels, la corrélation avec un pas de 1 pixel effectuée sur la totalité de l'objet prend 6 secondes ! ...

D'où l'idée de ne "lancer" la corrélation que dans les zones de probable concordance afin d'accélérer considérablement le traitement.

V.2.3. Méthode retenue : Combinaison heuristique de V.2.1 et V.2.2

L'idée consiste à combiner les avantages des deux méthodes précédentes, à savoir :

- la grande rapidité du tri des caractéristiques globales,
- la grande précision de la méthode de corrélation sur les signatures.

Voyons comment fonctionne cet algorithme :

On calcule le nombre de coïncidences entre les gros segments de l'objet inconnu et ceux des objets de référence. On constitue donc un tableau de coïncidences basées sur le principe défini en V.2.1.

Indice du gros segment de l'objet de référence pris en correspondance avec le premier gros segment de l'objet inconnu.

		1er gros segment	2ème gros segment	——	Ième gros segment
		Nbre de coïncidences			
Objet de référence	1er objet	X_{11}	X_{12}	——	X_{1I}
	2ème objet	X_{21}	X_{22}	——	X_{2I}
	néme objet	X_{n1}	X_{n2}	———	X_{nI}

Dans chaque colonne apparait le nombre de coïncidences relevées, appelées notes de coïncidences.

A la fin de ce traitement, on effectue la corrélation entre l'objet de référence ayant eu le plus grand nombre de coïncidences, l'abscisse prise comme orgine sur l'objet inconnu est l'abscisse du premier gros segment ; l'abscisse prise comme origine sur l'objet de référence est celle du gros segment pris en correspondance. On n'effectue cette corrélation que pour un petit nombre de décalages (par exemple plus ou moins le vingtième du périmètre).

Si, à l'issue de ce calcul, la distance minimale trouvée est trop importante, on prend dans le tableau de caractéristiques la "note" immédiatement inférieure si elle reste néanmoins acceptable (supérieure à une valeur minimum) et l'on recommence la corrélation avec le nouvel objet de référence et la nouvelle abscisse d'origine.

Cette méthode a donné d'excellents résultats : elle est aussi performante que si l'on avait effectué la corrélation totale avec tous les objets de référence. Le temps de reconnaissance, pour le même objet que précédemment devient inférieur à 1 seconde !... Nous avons, grâce à ces heuristiques, gagné un facteur temporel de 60.

V.3. RESULTATS

Nous allons exposer les résultats des trois méthodes ainsi que leur temps de calcul.

Nous avons fait l'apprentissage de 10 formes et chaque algorithme examine la possibilité de classement de l'objet présenté.

Les formes proposées sont les suivantes :

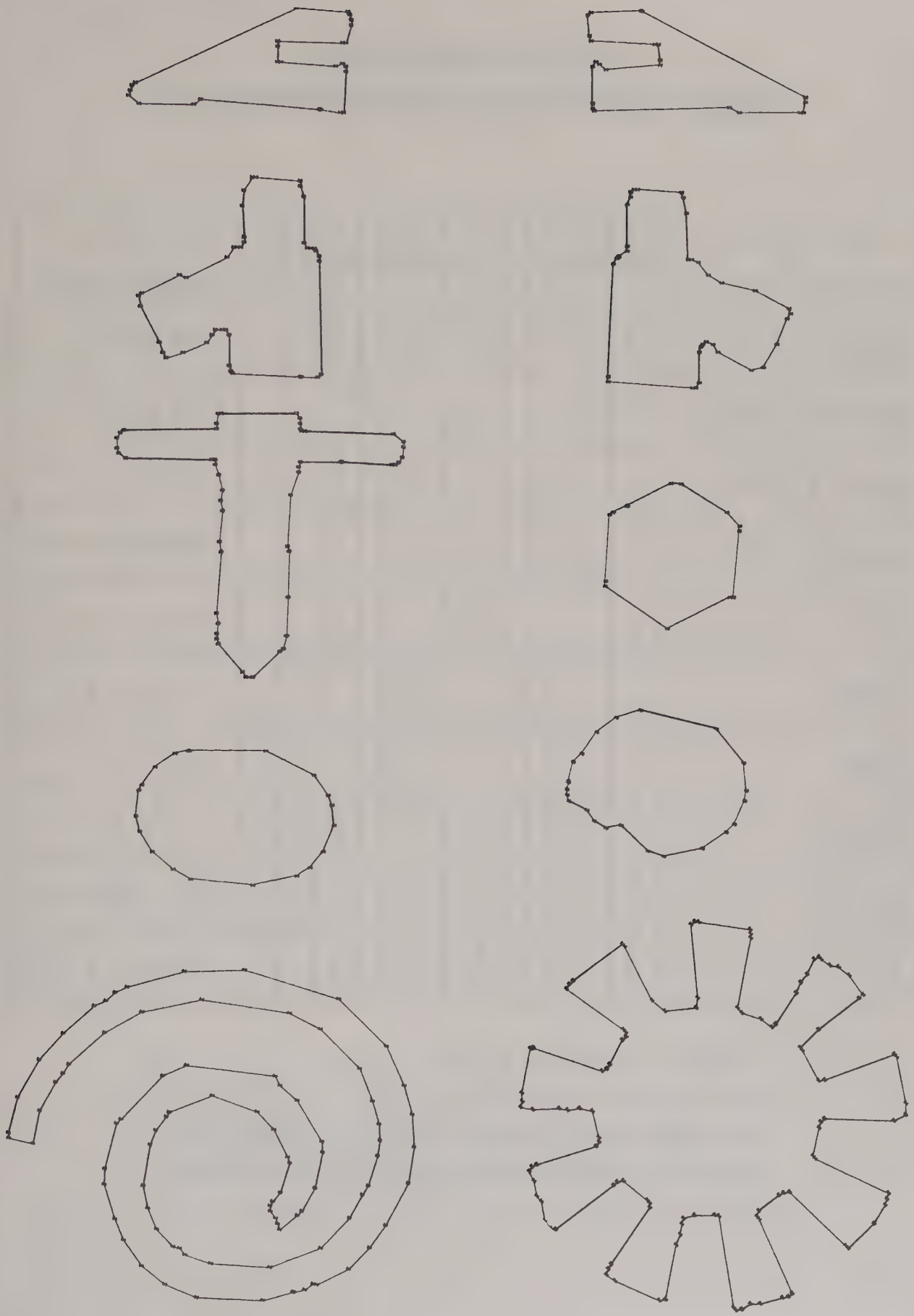


Figure 41 : Les objets à reconnaître

TABLE DE CONFUSIONS

Méthode testée : tri par caractéristique

	MD	MG	RD	RG	C	E	O	F	S	B	Temps moyen de reconnaissance
Mortaise droite	8	1		1							≠ 0,2 s
Mortaise gauche		10									≠ 0,2 s
Raccord droit			10								≠ 0,2 s
Raccord gauche			1	9							≠ 0,2 s
Clé					10						≠ 0,2 s
Ecrou						8	2				≠ 0,2 s
Ovale							8			2	≠ 0,2 s
Fleur								10			≠ 0,5 s
Spirale									10		≠ 0,5 s
Blob						1	2			7	≠ 0,2 s

. 5 objets reconnus à 100%
. Taux moyen de réussite : 90%
. Le temps de traitement dépend du nombre de caractéristiques retenues par le programme

TABLE DE CONFUSIONS

Méthode testée : corrélation pas de 1 pixel

	MD	MG	RD	RG	C	E	O	F	S	B	Temps moyen de reconnaissance
Mortaise droite	10										≠ 50 s
Mortaise gauche		10									≠ 50 s
Raccord droit			10								≠ 50 s
Raccord gauche				10							≠ 50 s
Clé					10						≠ 50 s
Ecrou						10					≠ 50 s
Ovale							10				≠ 50 s
Fleur								10			≠ 75 s
Spirale									10		≠ 70 s
Blob										10	≠ 50 s

. 10 objets reconnus à 100%
. Taux moyen de réussite : 100%
. Le temps est sensiblement constant car on dilate l'objet inconnu afin qu'il aie le même périmètre que l'objet de référence testé.

TABLE DE CONFUSIONS

Méthode testée : combinaison heuristique

	MD	MG	RD	RG	C	E	O	F	S	B	Temps moyen de reconnaissance
Mortaise droite	10										1,3 s
Mortaise gauche		10									0,8 s
Raccord droit			10								0,85 s
Raccord gauche				10							0,8 s
Clé					10						1 s
Ecrou						10					1,1 s
Ovale							10				1 s
Fleur								10			2,3 s
Spirale									10		2,1 s
Blob										10	0,9 s

. 10 objets reconnus à 100%
. Taux moyen de réussite : 100%
. Les grandes variations de temps s'expliquent par le fait qu'il dépend des succès/échecs lors des tentatives de corrélation.

V.4. CONCLUSION

Nous avons exposé dans ce chapitre deux méthodes de reconnaissance très différentes sur le plan du principe.

Après avoir montré les avantages et inconvénients de ces méthodes, nous avons étudié une heuristique combinant leurs avantages : on obtient ainsi un taux moyen de réussite de 100% avec un temps moyen de 1,25 seconde pour la reconnaissance (il faut bien sûr ajouter le temps lié à l'extraction du contour et à la segmentation).

PARTIE C

UNE APPLICATION INDUSTRIELLE

CHAPITRE VI : LE SYSTEME ANIMA

VI.1. PRESENTATION

ANIMA est un système de vision évolutif destiné essentiellement à des applications en robotique. Ce système est capable, à partir d'une image peu contrastée d'identifier une forme d'objet d'après sa silhouette et de l'associer à une classe. Les classes d'objets sont définies au préalable avec l'aide de l'opérateur au cours d'une période d'apprentissage. Le système peut, en outre, dans le mode de reconnaissance :

- indiquer l'orientation de la pièce,
- situer des points caractéristiques (par exemple les points de prise) sur la pièce,
- donner une estimation de la déformation du contour de l'objet par rapport à celui enregistré dans la classe retenue.

La conception d'ANIMA est modulaire, c'est à dire que l'utilisateur peut choisir la configuration logicielle adaptée à son besoin particulier.

ANIMA peut être autonome (par exemple pour effectuer de l'inspection automatique d'objets) ou relié à un système logiciel extérieur et constituer ainsi l'unité de vision d'un système plus complexe (par exemple : capteur de haut niveau d'un robot). Dans ce but, nous avons décidé d'adopter le standard de la société GIXI, filiale du CEA. Cette société fabrique des systèmes robotiques "clé en mains" et a développé un langage de commande proche du BASIC : ROL qui dispose de fonctions de haut niveau liées à la robotique.

Le système ANIMA est destiné à être rattaché à cette gamme de produits en étant compatible tant sur le plan matériel que logiciel.

VI.2. DESCRIPTION MATERIELLE DU SYSTEME

VI.2.1. L'ensemble matériel est constitué de trois cartes au standard MULTIBUS largement diffusées sur le marché :

- une carte processeur assurant la totalité du traitement INTEL SBC 86-12 disposant de 64 K octets de RAM et 16 K octets de PROM,
- une carte mémoire d'image MATROX RGB256/4 de 256 x 256 x 4 bits,
- une carte digitaliseur vidéo de 4 bits MATROX FG01/4

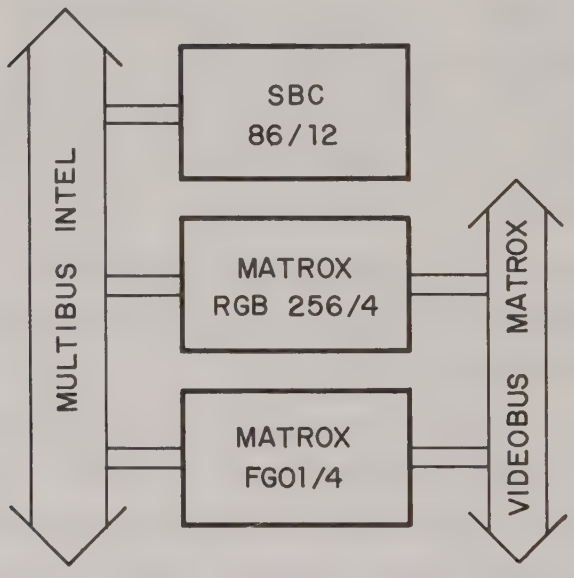


Figure 42 _ Structure matérielle.

VI.2.2. Configuration minimale

Les trois cartes peuvent être insérées dans un châssis INTEL SBC 660. Les commandes de contrôles interface se limitent à :

- un clavier alpha-numérique à liaison série RS 232 permettant à l'opérateur d'envoyer des ordres au système,

- une caméra au standard vidéo européen,
- une console vidéo permettant de restituer à l'opérateur :
  . le contrôle direct de l'image acquise par la caméra
  . le contrôle du contenu de la mémoire d'image
  . l'affichage de messages alpha-numériques.

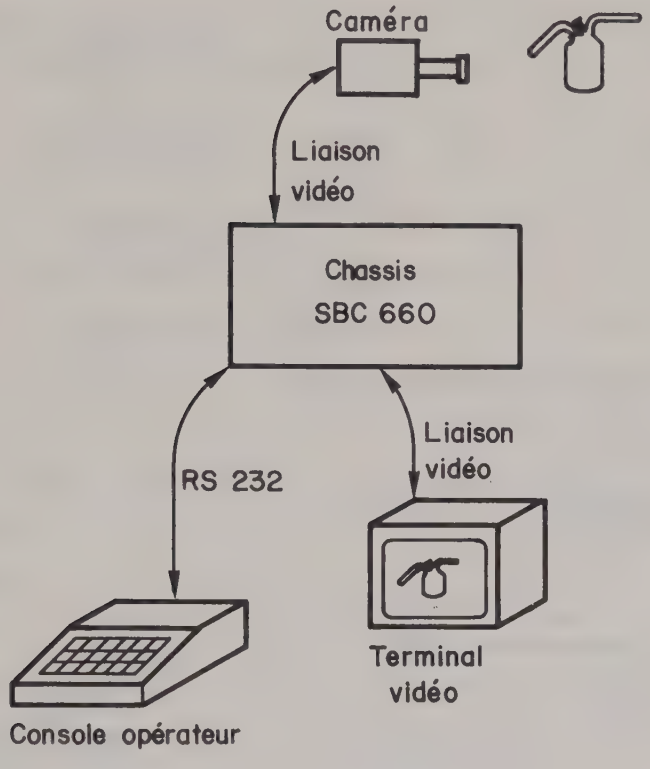


Figure 43_ Configuration minimale.

VI.2.3. Extensions

On peut, en rajoutant les modules de logiciel appropriés :

- étendre le nombre de caméras à 4,
- adjoindre une console alpha-numérique ou une console graphique de type TEKTRONIX 4014 par la liaison série RS 232,
- adjoindre un terminal couleur au standard PAL et visualiser en pseudo-couleurs le contenu de la mémoire d'image,

- adjoindre un crayon lumineux (module logiciel non réalisé),
- raccorder le système à d'autres processeurs 8086 compatibles MULTIBUS grâce à la structure logicielle d'ANIMA qui est compatible avec les moniteurs temps réel multi-tâches MMT 86 et RMX 86.

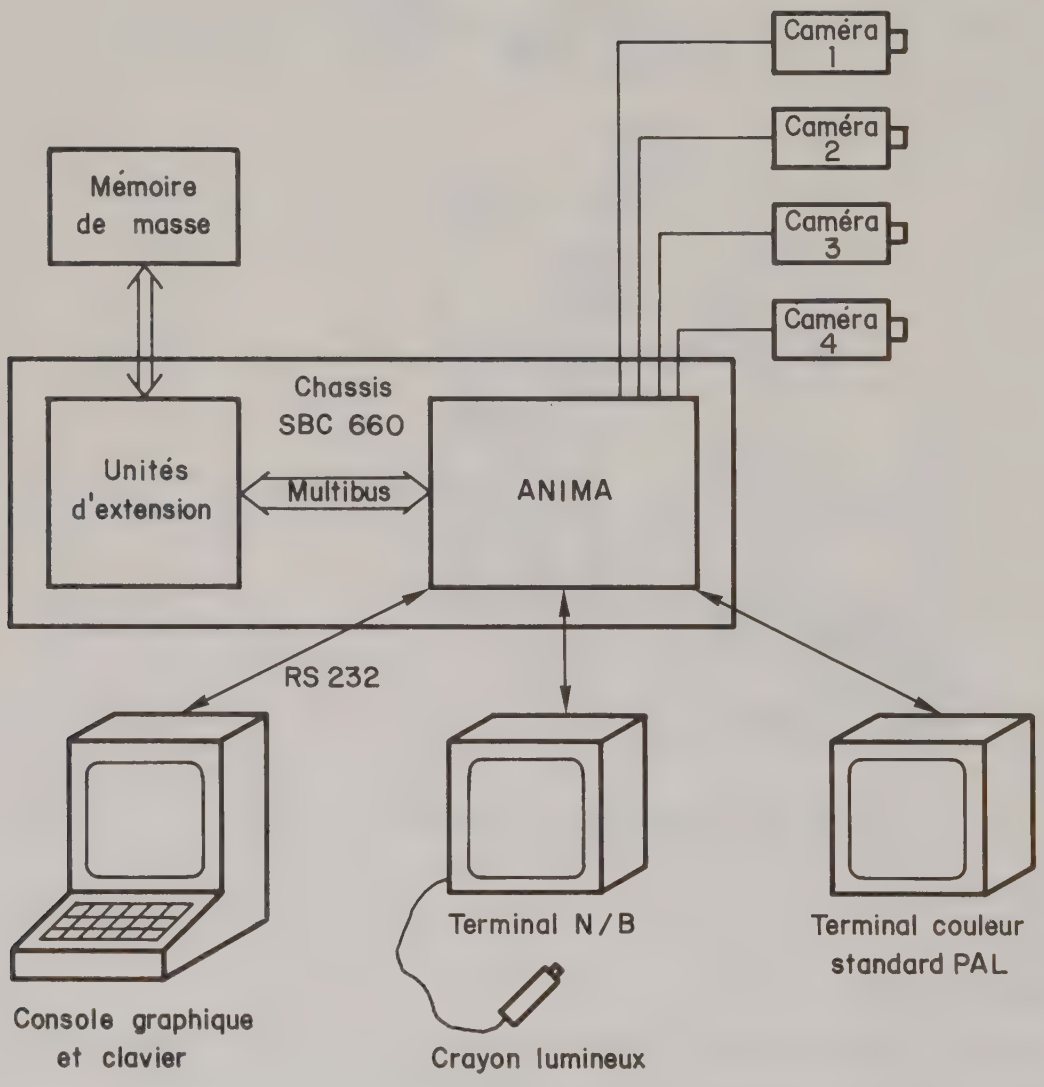


Figure 44 _ Une configuration possible du système.

(On pourra aussi, ultérieurement, étendre la configuration de la mémoire d'image à 256 x 256 x 8 bits en modifiant le digitaliseur et en rajoutant une carte de mémoire d'image RGB 256/4).

VI.3. CONFIGURATION LOGICIELLE DU SYSTEME

VI.3.1. Moniteur multitâche temps réel : MMT 86 ou RMX 86

La configuration logicielle d'ANIMA a été rendue compatible avec deux moniteurs de gestion multi-tâches afin de pouvoir adapter ce système à n'importe quelle unité informatique utilisant ces moniteurs.

* Le MMT86 est un moniteur temps réel multi-tâches étudié par la société GIXI pour le micro-processeur 8086 de INTEL. Ce moniteur permet de traiter en temps réel un certain nombre d'événements (exemple : fonction de transmission ou fonction d'acquisition cyclique), tout en permettant d'effectuer en différé le traitement des informations recueillies.

Le moniteur MMT86 peut gérer des applications nécessitant de telles possibilités et peut résoudre les problèmes suivants :

- Gestion d'événements aléatoires ou interruptions, les traitements associés étant effectués par des tâches dites "matérielles". Les niveaux d'interruptions sont configurables dans le module-source, ainsi que le nombre de ces interruptions (en standard 8 niveaux d'interruption, le niveau le plus prioritaire (n Ø) correspondant à l'interruption n 8 - INTEL).

- Il peut synchroniser des tâches (matérielles ou logicielles) par priorités ou par événements internes.

- Il peut gérer des ressources par exclusion mutuelle entre tâches.

- Il peut gérer des files d'attente de messages entre tâches.

- Enfin, il gère des délais pour activation d'événements ou inclusion de messages dans une file d'attente.

Le MMT86 se présente sous la forme de deux modules sources représentant au total 2700 octets de code. Il a besoin pour son fonctionnement interne de 550 octets de RAM, non compris le pool de délais (un buffer du pool fait 14 octets, donc avec un maximum de 32 délais, le moniteur a besoin au total de 1 K de RAM).

* Le RMX86 est un moniteur temps-réel multi-tâches, multiprogrammation étudié par INTEL pour le micro-processeur 8086. Ce moniteur, beaucoup plus important en volume logiciel, permet, en plus des fonctions assurées par MMT86 :

- la multiprogrammation,
- la gestion de mémoire de masse.

VI.3.2. Architecture générale

Du point de vue logiciel, le moniteur temps-réel gère l'enchaînement des tâches par l'intermédiaire du système de communication.

Les différents sous-systèmes, sous son contrôle, sont :

- l'interface avec l'opérateur,
- le sous-système de programmation permettant de faire l'interprétation des commandes écrites en ROL,
- le sous-système de vision ANIMA,
- les sous-systèmes d'extension.

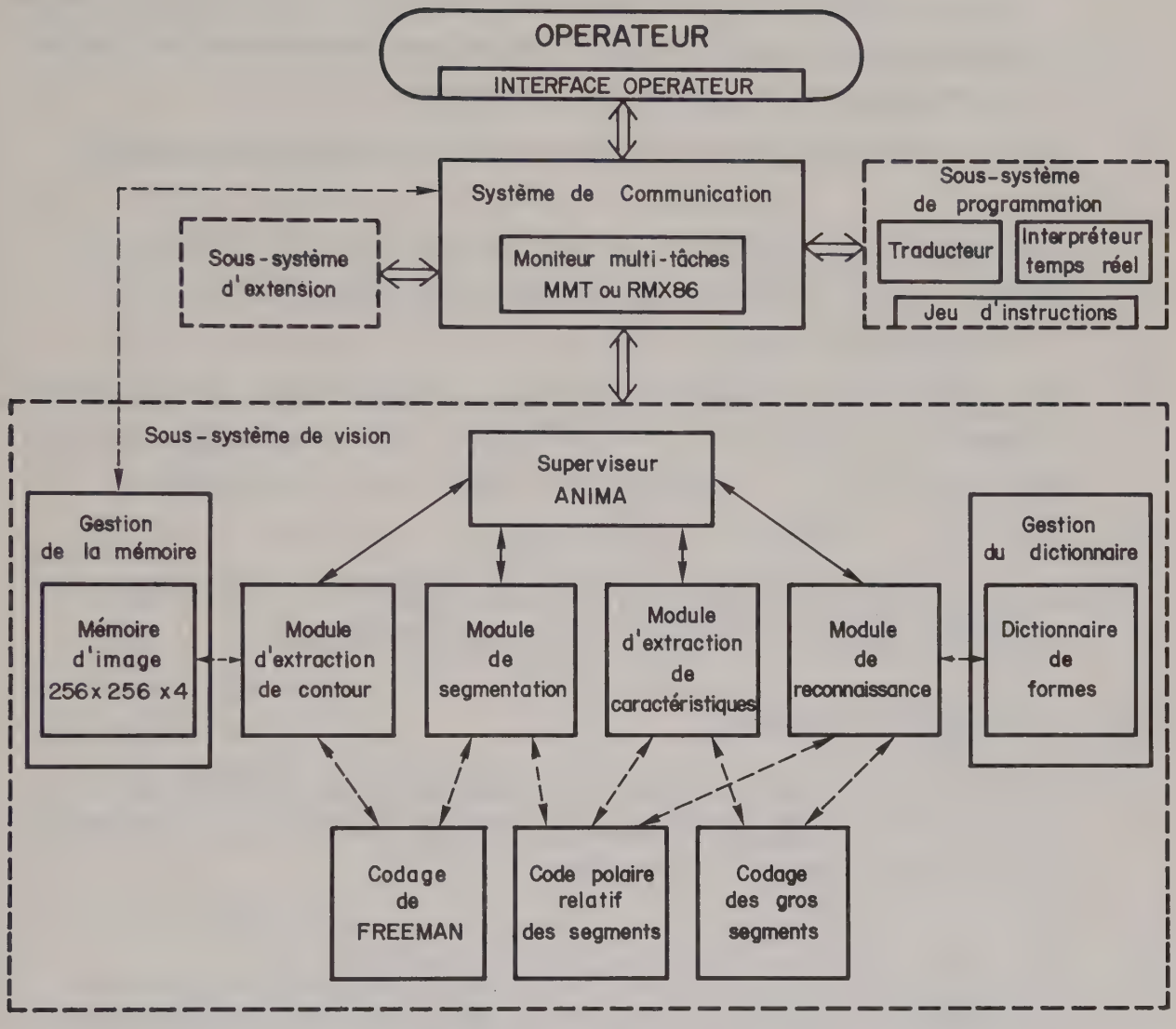


Figure 45 _ Structure logicielle.

L'opérateur dialogue avec le système de communication et peut :

- Envoyer des commandes aux différents modules, si ces commandes sont écrites en ROL, elles sont envoyées au sous-système de programmation. Sinon, elles peuvent être directement destinées aux autres sous-systèmes à l'aide d'un langage de commande spécifique.
- Recevoir des informations sur l'état des opérations.

La requête de l'opérateur est toujours prioritaire ; il peut, à tout moment, interrompre le déroulement de la commande en cours.

Le superviseur d'images assure le déroulement de la tâche demandée et appelle les différents modules nécessaires à son exécution en fonction des problèmes rencontrés au cours du traitement.

Examinons le rôle des modules en question, ainsi que les possibilités de substitutions envisagées.

VI.3.3. Les modules d'ANIMA

L'utilisateur disposera d'une panoplie de modules qui lui permettra de définir son système logiciel en fonction de ses besoins et contraintes.

VI.3.3.1. Module d'extraction de contour

Ce module réalise l'algorithme de suivi dynamique de contour décrit en III.3.

Deux fonctions principales lui sont allouées :

- une fonction indiquant la présence d'un contour coupant une ligne paramétrée de l'image (fonction LOOK),
- une fonction réalisant le suivi du contour détecté par LOOK (fonction RE-LOOK).

Partant des points de la mémoire d'image, ce module code les contours selon Freeman et stocke les coordonnées cartésiennes des points de départ des chaînes.

L'inconvénient majeur de ce module est sa durée d'exécution. C'est pourquoi, l'utilisateur disposant des conditions d'éclairage permettant la binarisation de l'image par seuillage pourra substituer à ce module un module d'extraction de contour sur images binarisées.

Ce module réalisera le seuillage de l'image et appliquera sur celle-ci un opérateur de connexité très rapide.

Le codage résultant sera identique à celui donné par le module original.

Les deux modules seront donc interchangeables. On espère un gain en temps d'un facteur 10 mais il est clair qu'alors le contraste de l'image devra être excellent.

L'utilisateur pourra donc choisir entre la souplesse et la rapidité...

VI.3.3.2. Module de segmentation

A partir de la table des codes de FREEMAN, ce module réalise la segmentation linéaire du contour d'après la méthode décrite en IV.1, puis effectue le codage invariant exposé en IV.2.

A la fin de ce traitement, on dispose des coordonnées polaires relatives des extrémités de segments.

VI.3.3.3. Module d'extraction de caractéristiques

Ce module réalise la méthode décrite en IV.3. Les gros segments caractéristiques trouvés sont enregistrés dans une table de caractéristiques.

VI.3.3.4. Module de reconnaissance

L'utilisateur dispose de deux options :

- un module effectuant un tri rapide sur les contours, mais ne vérifiant pas la pertinence du contour : version rapide. Il s'agit de l'algorithme développé en V.2.1.,

- un module effectuant en plus l'inspection sur le contour : version rigoureuse mais lente. Il s'agit de l'algorithme développé en V.2.3.

Le rapport entre les temps d'exécution est environ 5 (voir V.3.).

VI.3.3.5. Gestion de la mémoire d'image

Ce module réalise les entrées-sorties dans l'image. Il gère l'acquisition d'une image et l'effacement de l'écran.

Il dispose en plus d'un générateur de caractères permettant d'afficher un texte alpha-numérique sur l'écran.

VI.3.3.6. Gestion du dictionnaire de formes

Ce module permet d'enregistrer sous un nom formé par 16 caractères alpha-numériques :

- les caractéristiques globales,
- le codage polaire relatif,

d'une forme dans la mémoire vive du système ce module permet l'accès au dictionnaire, l'effacement ou la remise à jour de formes.

VI.4. UTILISATION

L'opérateur dialogue avec le système à l'aide du langage évolué ROL. Une instruction typique est de la forme :

[Etiquette]	mot clé	paramètres
	CODE FONCTION	PARAMETRES CLE DE LA FONCTION

Ce langage, assimilable à un BASIC simplifié est étudié pour les applications en robotique : l'utilisation de mots-clés représentant des fonctions élaborées rend ce langage très souple et aisé d'emploi.

En ce qui concerne la vision, certains mots-clés vont être ajoutés au langage. Actuellement, ces mots-clés n'ont pas été fixés définitivement. Nous publions donc une proposition de syntaxe, ainsi que les noms actuellement employés.

Chaque instruction exécutée envoie, en fin d'exécution, un compte-rendu (mot de 16 bits) indiquant s'il y a eu des problèmes au cours de l'exécution. Ce code compte-rendu peut aiguiller sur un programme d'erreur et enchaîne ainsi les opérations nécessaires.

Certaines instructions demandent un délai lié au temps d'accès à la mémoire d'image. Le moniteur peut alors mettre en sommeil la tâche pendant le délai correspondant.

FONCTIONS PROPRES AU TRAITEMENT D'IMAGES

FONCTIONS	Nom proposé	Paramètres		Compte-rendu	Délai	Remarques
		Envoyés	reçus			
Initialisation	INIT-VISU	0	0	0 = Init reçue	OUI	Initialise le système de vision. Une mire apparait sur l'écran
Caméra	CAMERA	Numéro de caméra	0	0 = caméra sélectée	NON	Définit la voie à inspecter (de 0 à 3)
Acquisition	PICTURE	0	0	0 = Acquisit demandée	OUI	Fait une acquisition de l'image
Effacement de l'écran	CLEAR	(niveau de gris)	0	0 = Effacement demandé	OUI	L'écran est mis uniformément à la valeur donnée
Sortie de texte	BLA	Chaîne de caractères - X - Y - taille - Niveau de gris	0	0 = Texte affiché	NON	Affiche le texte sur l'écran au point de départ (X,Y)
Cherche	LOOK	- Seuil - N° de ligne	0	0 = contour présent 1 = contour absent	NON	Exploration de la ligne donnée en paramètre dès que l'on détecte un contour de longueur $l > L_{min}$, retour au moniteur
Relance	RE-LOOK	0	0	0 = contour fermé 1 = contour non fermé FFH = erreur	NON	Relance l'extraction du contour. On ne peut appeler RE-LOOK qu'après avoir appelé LOOK
Trace	TRACE	- niveau de gris du tracé	0	0 = tracé fait 2 ou 3 = erreur de chaînage	NON	Le tracé apparaît sur le moniteur avec le niveau donné en paramètre
Extrapolation	EXTR	0	0	0 = Extrapolation faite	NON	Effectue la segmentation du contour

FONCTIONS	Nom proposé	Paramètres		Compte-rendu	Délai	Remarques
		Envoyés	reçus			
Sélection des points de prise	HOLD-S	0	0	0 = sélection faite	OUI	L'opérateur sélectionne sur le contour les points de prise
Codage	ENCODE	0	0	2 = erreur chaînage 0 = codage fait 1 = contour non fermé	NON	Effectue le codage polaire relatif du contour
Calcul des caractéristiques	CHARAC	- Ro max - Theta max	diamètre du cercle	0 = codage effectué 1 = contour non fermé 7 = cercle		Calcule les caractéristiques globales du contour

FONCTIONS PROPRES A LA GESTION DU DICTIONNAIRE DE FORMES

FONCTIONS	Nom proposé	Paramètres		Compte-rendu	délai	Remarques
		Envoyés	Reçus			
Enregistrement d'une forme	RECORD	Nom (≤ 16 caractères)	0	0 = nom accepté 2 = dictionnaire plein 3 = nom invalide 6 = nom existant (remise à jour)	NON	Enregistre les caractéristiques de la forme dans le dictionnaire
Effacement d'une forme	DELETE	Nom (≤ 16 caractères	0	0 = forme effacée 3 = nom inconnu	NON	Efface la forme du dictionnaire
FONCTIONS DE RECONNAISSANCE						
Tri par caractéristiques	SORT	Nbre de maxima maximum	-Nom retenu -certitude -décalage -pointeur de la table de coïncidence	0 = tri effectué 1 = dictionnaire vide	NON	Fait le tri par caractéristiques
Corrélation	CORR	-chaîne de 16 caractères -origines des repères -seuil de divergence -longueur à correler	-point de prise -orientation -écart de tolérance	0 = corrélation faite 1 = débordement 3 = nom inconnu 7 = cercle	NON	Effectue la corrélation de la forme avec la forme spécifiée par la chaine de caractères -si "?" alors méthode automatique V.2.3 avec toutes les formes. -si "ALL" corrélation avec toutes les formes -si "nom" corrélation avec la forme spécifiée par "nom"

FONCTIONS DE HAUT NIVEAU

FONCTIONS	Nom proposé	Paramètres		Compte rendu	Délai	Remarques
		Envoyés	Reçus			
Apprentissage	LEARN	-Chaîne de caractères (nom)	0 Périmètre	0 = codage effectué nom accepté 1 = contour non fermé 2 = dico plein 3 = nom invalide 5 = anomalie : les pts de prises non sélectionnés 6 = nom existant (remise à jour) 7 = cercle	NON	fait EXTR ENCODE CHARAC si non cercle, RECORD
Reconnaissance	RECN	-seuil de tolérance	-Nom de l'objet -pts de prise -orientation -écart tolérance -nb de caract. reconnues	0 = objet reconnu 1 = ERREUR 7 = cercle	NON	fait EXTR ENCODE CHARAC SORT CORR automatique (V.2.3)

VI.5. UN EXEMPLE D'APPLICATION : SAISIE AUTOMATIQUE D'OBJETS SUR UN TAPIS ROULANT

VI.5.1. Nous décrivons sommairement un exemple d'utilisation du système de vision associé à la commande d'un bras de robot.

L'expérience consiste, lors d'une étape d'apprentissage, à introduire les caractéristiques de 10 formes d'objets dans le dictionnaire de formes.

Le système doit ensuite reconnaître ces objets arrivant sur un tapis roulant , arrêter le tapis, saisir l'objet et adopter la stratégie de mouvement définie pour la manipulation de cet objet (par exemple : positionner l'objet sur un autre tapis roulant, assembler cet objet avec un autre objet, ...).

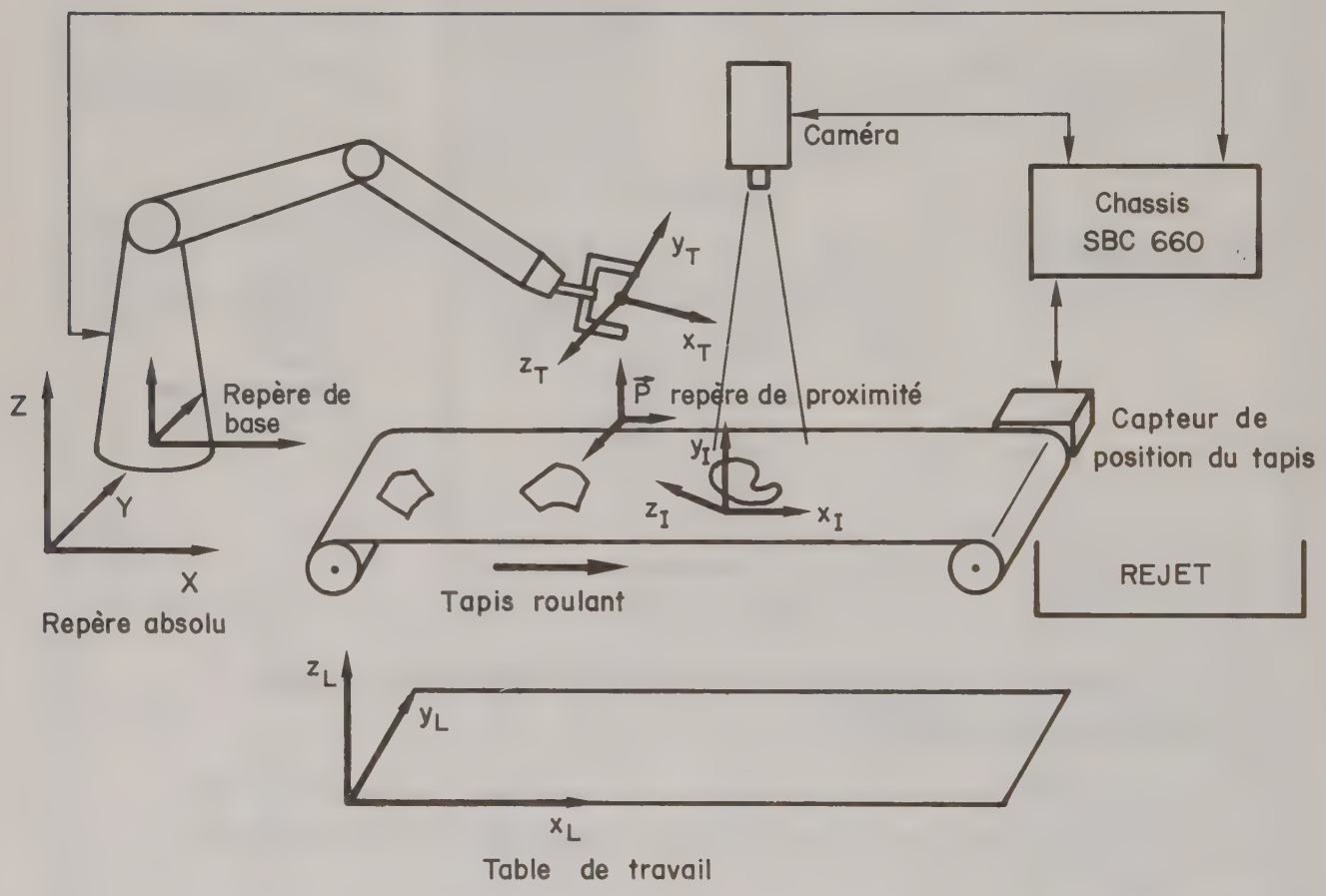


Figure 46 _ Une expérience de saisie automatique d'objets par un robot.

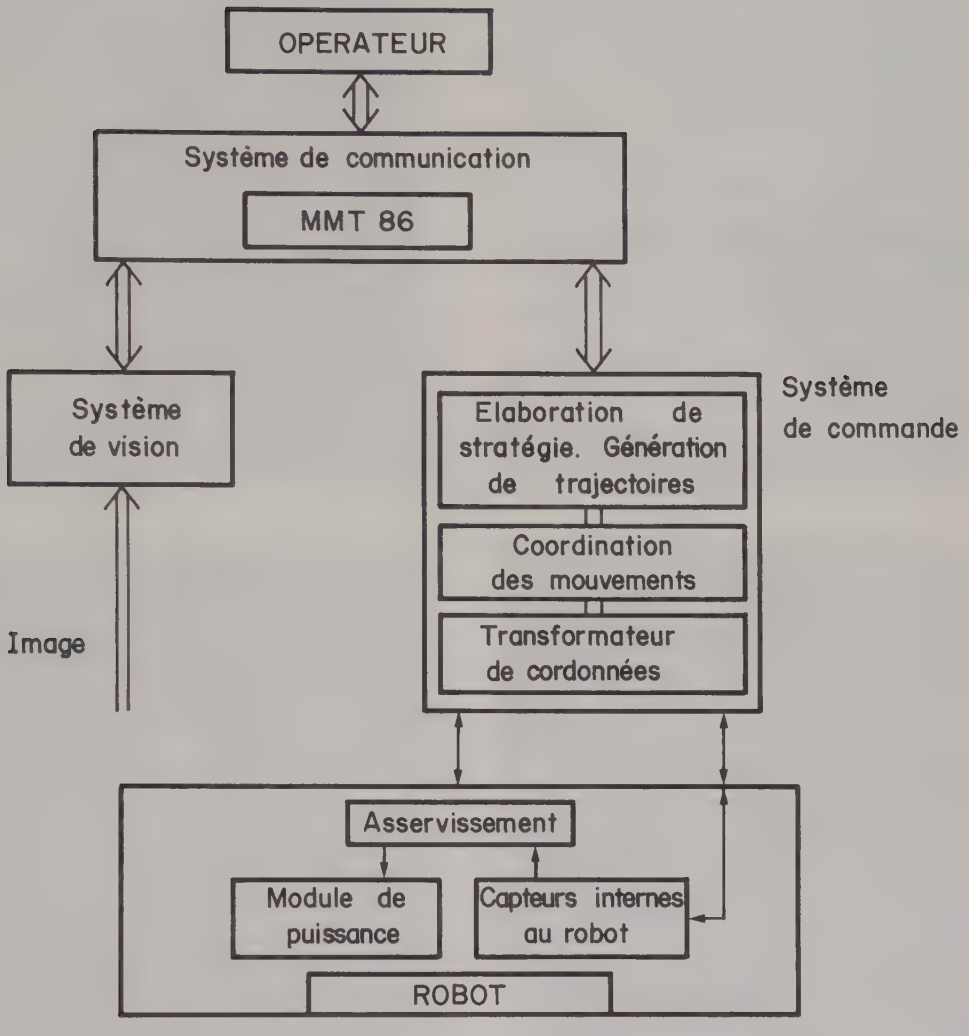


Figure 47_ Configuration logicielle du système

VI.5.2. Apprentissage

Au cours de cette phase, l'opérateur doit effectuer le "calibrage" du système en définissant l'origine et l'orientation du repère lié à l'image (x_I, y_I, z_I) par rapport au repère lié au corps du robot, que nous appellerons repère de base. On définit aussi un rapport d'échelle entre les deux repères (quelle est la taille réelle d'un point de l'image ?). Cette opération génère une fonction de transcription permettant de calculer les coordonnées d'un point dans un repère connaissant les coordonnées dans l'autre.

B = TRIB (A) conversion repère image → repère de base
B = TRBI (A) conversion repère de base → repère image

Après cette opération, on présente au système chaque objet pris dans toutes les positions d'équilibres possibles. L'opérateur détermine interactivement les points de prise de l'objet :

```
FUNCTION APPRENTISSAGE

10   PICTURE,          /* prendre une image (cf. fig. 48a)*/
     LOOK, 128     /* explorer ligne 128, y a t'il un
                      contour ? */
     IF CPTR = 1 THEN GO TO 10/* si pas de contour, aller
                                 en 10)*/
     RE_LOOK          /* extraire le contour */
     IF CPTR = 1 THEN GO TO 10/* si le contour n'est pas
                                 fermé, retourner en 10 */
     TRACE  16       /* tracé du contour (cf. fig. 48b)*/
     PRISE           /* l'opérateur sélectionne sur l'écran
                        les points de prise (cf. fig. 48d)*/
     INPUT $CHAINE /* l'opérateur entre le nom de l'objet*/
     LEARN $CHAINE /* codage de la forme et stockage dans
                      le dictionnaire */

     END APPRENTISSAGE.
```

On suppose à ce stade que toutes les opérations d'apprentissage liées au robot sont effectuées, à savoir :

- le calibrage des repères de travail,
- la stratégie à adopter en fonction des objets se présentant.

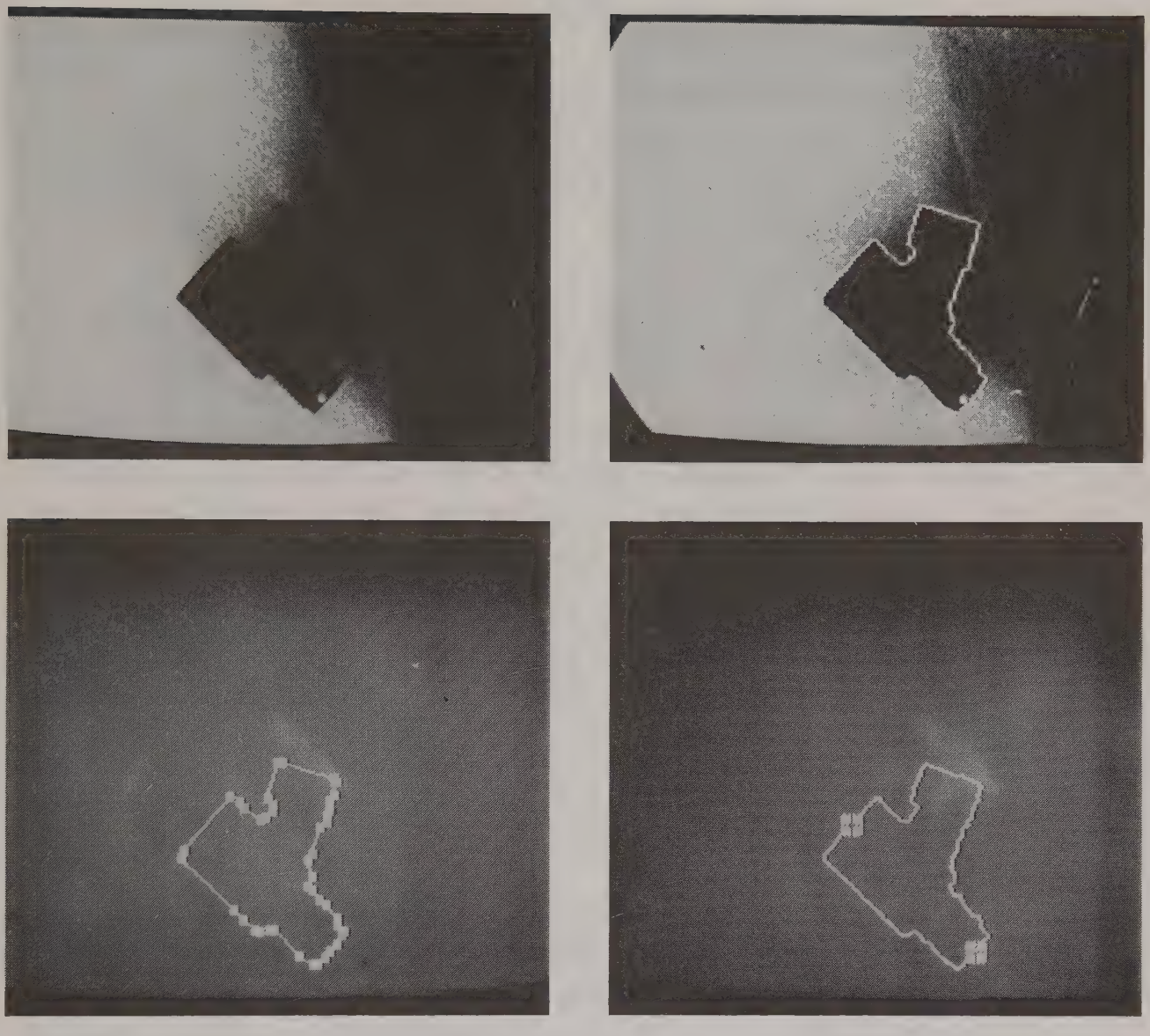


FIGURE 48 : Apprentissage

A	B
C	D

a) Image présentée
b) Tracé du contour
c) Segmentation du contour
d) Sélection par l'opérateur des points de prise

VI.5.3. Reconnaissance

Le module de commande du robot doit connaître :

- le nom de l'objet,
- les coordonnées des points de prise dans le repère de base,

afin de pouvoir générer la trajectoire désirée.

Dès qu'un objet se présente dans le champ de la caméra, on arrête le défilement du tapis roulant et on commence l'analyse de la forme.

Pendant ce temps, le module de commande du robot peut coordonner le déplacement de la pince vers un repère d'approche P situé à proximité du repère de l'image.

Dès que l'objet est reconnu, on envoie au module de commande les informations nécessaires pour prendre l'objet. Si l'objet est rejeté, on le laisse sur le tapis (on n'envoie pas de commande au robot et on relance la mise en marche du tapis.

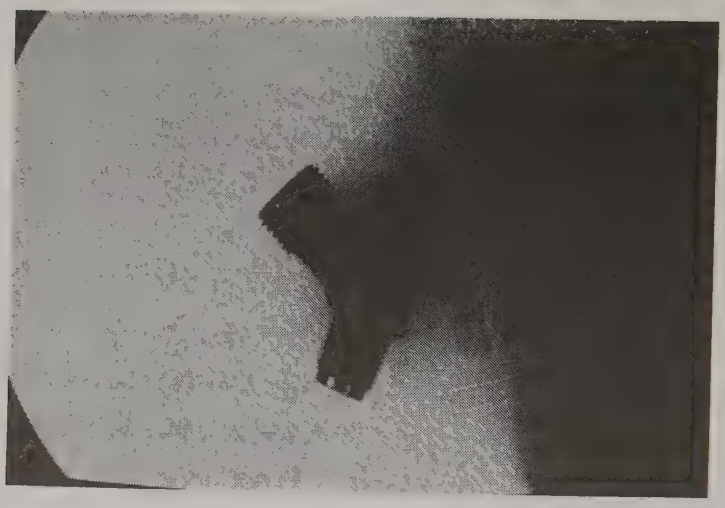
Si le module de commande du robot décèle des anomalies lors de la commande, il interrompra le processus en cours en appelant le moniteur.

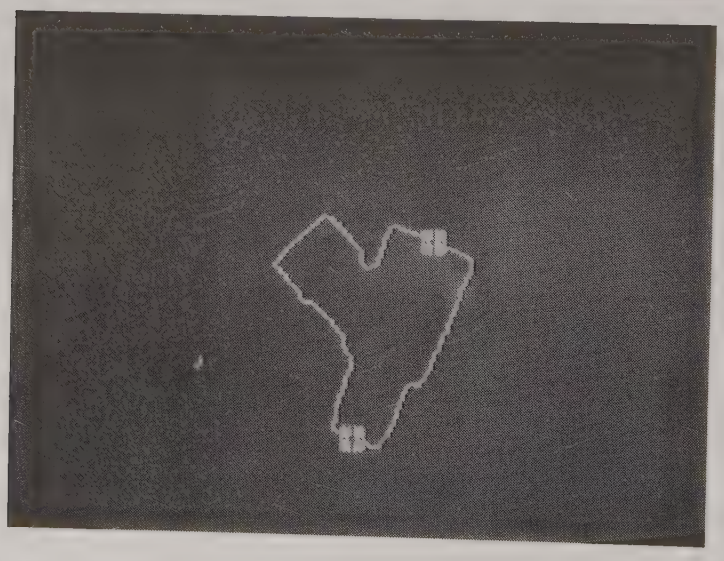
Un petit programme simple en ROL réalise cette fonction.

```
FUNCTION RECONNAISSANCE

1Ø    CONVEYOR = ON        /* Mettre en route le tapis */
2Ø    PICTURE              /* Prendre une image */
      LOOK   21Ø           /* On explore une ligne sur le bas
                              de l'écran */
      IF CPTR = 1  THEN GO TO 2Ø /* Si pas de contour présent,
                                    revenir en 20 */
      SEND (MODCOM, @ 1)   /* On indique au module de commande
                              la présence d'un objet : amener
                              le bras du robot en P */
      CONVEYOR = OFF       /* Arrêt du tapis */
      RE_LOOK              /* Extraire le contour */
```

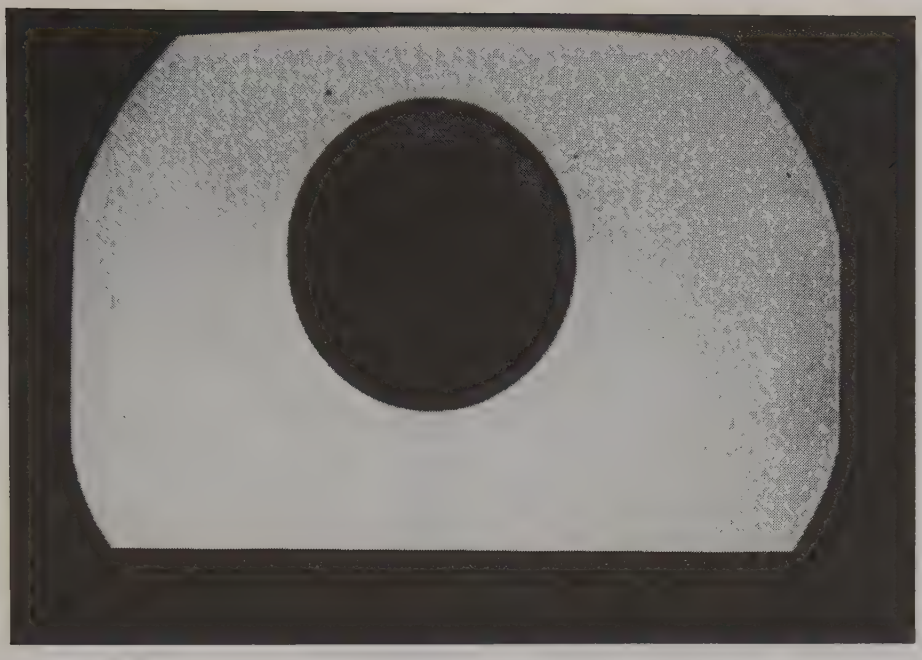
```
      IF CPTR = 1 THEN GO TO 1ØØØ /* Si le contour n'est pas
                                     fermé, rejeter */
      TRACE 16                    /* Tracer le contour */
      RECN 1ØØØ,@T                /* Reconnaître l'objet : les
                                     caractéristiques sont mises
                                     dans T */
      IF CPTR = 1 THEN GO TO 1ØØØ /* L'objet n'est pas connu
                                     → rejet */
      SEND (MODCOM,@T)            /* Envoi vers le module de commande
                                     des caractéristiques nécessaires
                                     à sa saisie */
      RECEIVE (MODCOM,@A)         /* Attente de message en provenance
                                     du module de commande */
      GO TO 1Ø                    /* La saisie a été correcte */
1ØØØ  CONVEYOR = ON               /* Attente d'un espace entre
                                     les objets */
      PICTURE
      LOOK 21Ø
      WHILE CPTR = Ø DO           /* Tant que l'on rencontre un
                                     contour, l'éliminer */
         PICTURE
         LOOK 21Ø
         ENDDO
      GO TO 2Ø                    /* On est sur un espace inter-
                                     objets → retour au début */
      END RECONNAISSANCE
```





A
B
C

FIGURE 49 : Reconnaissance
a) Image acquise
b) Contour extrait
c) Reconnaissance de l'objet et localisation des points de prise



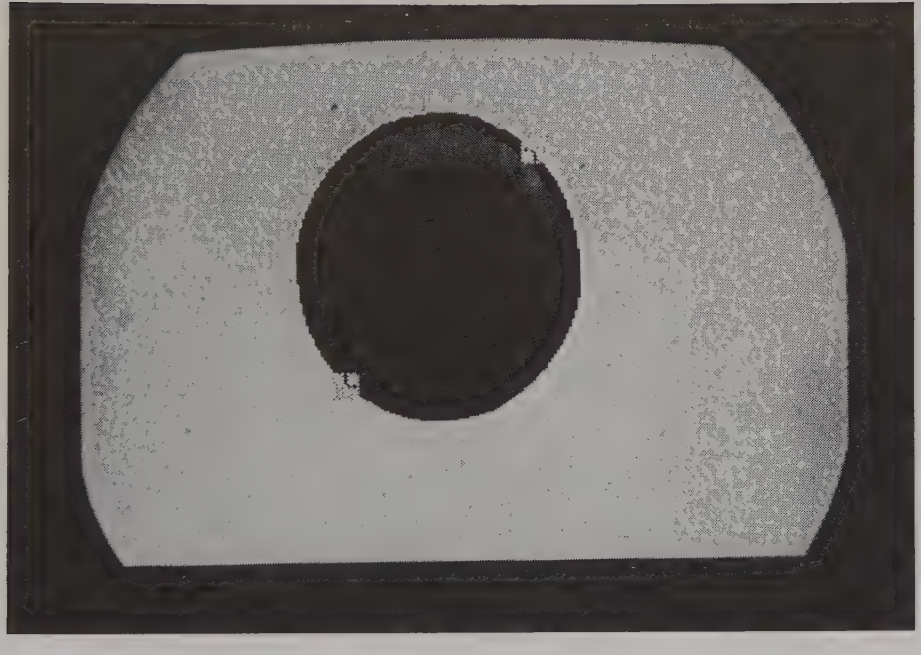


FIGURE 50 : Les objets circulaires sont reconnus directement d'après leur codage. Les points de prise sont arbitrairement calculés sur un diamètre.

A

B

On notera la possibilité de structuration du programme grâce aux instructions permettant des cycles : WHILE, DO ... ENDDO ; UNTIL, DO ... ENDDO ; IF THEN (ELSE), ENDIF ; CASE OF (ELSE).

Nous remarquons, par cet exemple, la simplicité de programmation et la concision du langage grâce aux fonctions de haut niveau accessibles par le langage ROL.

FIGURE 51 : Notre maquette d'études.

CONCLUSIONS

L'objectif qui a guidé ce travail a été d'améliorer l'adaptativité des robots par la maîtrise visuelle de l'intéraction environnement-robot.

Nous avons exposé, dans la première partie de ce mémoire, une méthode de suivi dynamique capable d'extraire les contours d'une image, évitant le traitement systématique de tous les points. Cette méthode utilise la description du contour par un parcours heuristique dans une arborescence de lignes, combinée à une fonction de coût quantifiant la qualité du contour détecté. L'originalité de cet algorithme réside dans sa bonne insensibilité au bruit : il permet d'obtenir des résultats satisfaisants, même dans des cas d'images de mauvaise qualité. Un codage du contour particulièrement adapté à cette méthode a été utilisé : le codage de Freeman.

Nous avons présenté ensuite une technique de compression de l'information par extrapolation linéaire de la ligne de contour. Les segments obtenus sont codés d'une manière invariante en rotation et translation, cette technique conserve les propriétes géométriques du contour.

Nous avons développé, dans la deuxième partie, une méthode de reconnaissance basée sur l'utilisation d'une distance entre formes calculée à partir du codage invariant. Cette distance quantifie la somme des écarts directionnels élémentaires relevés quand on parcourt simultanément les deux formes à comparer. Nous avons défini une "corrélation" entre ces deux codages bàsée sur l'utilisation de cette distance. Cette méthode a donné d'excellents résultats par leur précision, mais elle nécessite d'effectuer un volume de calculs incompatibles avec une reconnaissance en ligne. Il a donc été nécessaire d'accélérer considérablement cet algorithme en le "guidant" vers les zones de vraisemblables coïncidences à l'aide de notre algorithme de pré-classification rapide. Cette méthode de pré-classification est basée sur l'extraction de caractéristiques

grossières déduites du codage précédent : les "gros segments". Ces caractéristiques sont très pertinentes car elles correspondent à une description visuelle sommaire de la forme. Elles réduisent considérablement la quantité d'informations mais elles ne permettent pas de reconstituer le contour original. L'algorithme effectue un tri sur ces caractéristiques par comparaison à celles enregistrées dans un dictionnaire de formes et indique toutes les possibilités de superposition, qui seront reprises par l'algorithme de corrélation. Cette heuristique combine les avantages de précision et de rapidité des deux méthodes précédentes qui pourraient également être utilisées pour la reconnaissance. Nous avons testé avec succès cette heuristique de reconnaissance sur notre système et avons établi des tables de confusions pour dix objets testés avec les trois méthodes.

Notre étude a fait l'objet d'une application industrielle et va constituer le système de vision d'une gamme de robots.

Dès l'origine, nous avons orienté nos travaux vers le choix d'une solution logicielle et en langage évolué PL/M 86, en raison de la grande souplesse d'utilisation et du faible coût de mise en oeuvre. Le principal inconvénient résultant d'un tel choix est la relative lenteur d'exécution inhérente à l'aspect séquentiel du traîtement. Nous envisageons, dans un proche avenir, de réaliser en langage assembleur certains modules fréquemment appelés, notamment lors de l'extraction du contour, afin d'optimiser le temps du calcul et de réduire ainsi la durée du traitement.

Par ailleurs, l'avance technologique quotidienne permet de voir apparaitre sur le marché des processeurs de plus en plus performants aux temps de cycle-machine de plus en plus réduits : depuis le début de notre étude (Octobre 80), nous avons pu doubler la rapidité d'exécution en adoptant la nouvelle version du 8086 (8 Mhz).

Un prolongement intéressant de ces travaux consisterait en l'étude de structures câblées, suffisament générales pour ne pas imposer une rigidité excessive aux systèmes l'utilisant, adaptées à certains gros traitements spécifiques de données. Ces structures permettraient d'accélérer considérablement l'éxécution et rendraient possible, par exemple, le contrôle visuel en fin de chaine aux cadences généralement imposées par la F.A.O. (10 objets à reconnaître par seconde).

Le principal "goulot d'étranglement" de notre système apparaît au cours du pré-traitement des données : une structure spécialisée adaptée à notre méthode d'extraction de contours consisterait en un réseau de processeurs câblés selon la structure arborescente définie dans l'image (cf figure 15).A chacun des points à examiner dans notre graphe limité à quatre générations, correspondrait un processeur spécialisé qui aurait pour tâche de calculer le côut de succession entre son(ou ses) propre(s) père(s) et lui-même. Une telle structure est très intéressante à réaliser : elle réduit considérablement l'aspect séquentiel du parcours de graphe et n'impose pas de contraintes sur la stratégie de poursuite, qui reste entièrement programmable. Il suffit de charger les 24 points de l'image correspondant à la zone locale à étudier dans la mémoire du réseau, qui calculera simultanément tous les coûts de succession du graphe. Un libre choix est laissé à l'utilisateur de la structure pour interpréter ces données et pour décider du parcours à adopter.

Le grand nombre d'opérations à effectuer lors de l'inspection finale du contour par la corrélation constitue le deuxième goulot d'étranglement de notre système. Notre représentation codée du contour en "signatures" peut être assimilée à un signal mono-dimensionnel, nous pouvons envisager

d'adapter des opérateurs câblés de convolution conçus pour les traitements numériques du signal afin d'accélérer le calcul de la corrélation. Cependant, une grande prudence s'impose dès que l'on désire concevoir une structure spécialisée en traitement d'images et nous envisageons, dans un premier temps, de nous limiter à une étude de faisabilité de ces structures très onéreuses à développer.

Nous pensons que les années à venir laisseront une part de plus en plus importante à la vision dans la communication homme-machine. Il est donc nécessaire que la machine sache non seulement synthétiser une image, mais aussi percevoir le monde extérieur par l'image. De nombreux chercheurs travaillent dans cette voie et déjà plusieurs systèmes de vision sont commercialisés et opérationnels dans certains ateliers flexibles modèles. Notre travail s'inscrit dans cette démarche générale et constitue, par ses résultats positifs et concrets, un nouvel apport à ce sujet ouvert qu'est la reconnaissance automatique des images appliquées à la robotique.

BIBLIOGRAPHIE

/1/ C. PREVOST
Reconnaissance artificielle des formes sur les méthodes de détection de contours.
Le Nouvel Automatisme - Mars 1982

/2/ A. ROSENFELD - M. TAVAKOLI
Edge segment linking based on grey level and geometrical compatibilities.
Pattern Recognition - Vol. 15 - 1982

/3/ S.T. BARNARD
Automated inspection using grey-scale statistics
S.R.I. 5th International Conference on Pattern Recognition - 1981

/4/ R.P. KRUGER - W.B THOMPSON
A Technical and economic assessment of computer vision for Industrial Inspection and Robotic assembly
Proceedings of the IEEE - Vol 69 - N° 12 - Dec. 1981

/5/ M. YACHIDA - S. TSUJI
A versatile machine vision system for complex industrial ports.
IEEE Trans-on computers - Sept. 1977

/6/ P. COIFFET
Les robots - Tomes I et II
Hermès - Sept. 1981

/7/ W.K. PRATT
Digital Image Processing
Willey Interscience - 1978

/8/ J.C. SIMON - A. ROSENFELD
Digital Image Processing and Analysis
Applied Science n° 20 - 1977

[9] C.Y. SUEN - R.DE MORI
Computer Analysis and Perception
CRC Press

[10] E. PERSOON
Shape description using Fourier transform
IEEE Systems

[11] A. NATO
Automatic interpretation and classification of images
A. Grasseli, Academic Press

[12] M. HUECKEL
An operator which locates edges in digital pictures
J.A.C.M. n° 18 - Janvier 1971

[13] T. PAVLIDIS
Structural pattern recognition
Springer Verlag Berlin - 1977

[14] T. PAVLIDIS
Curve fitting as a pattern recognition problem
6th conference on pattern recognition, Munich - Oct. 1982

[15] C.C. BRICE - C.L. FENNEMA
Scene analysis using regions
S.R.I. Artificial Intelligence - Volume 1 - Aut. 70

[16] A. MARTELLI
An application of heuristic search method to edge and contour detection
Commun, Ass Computer Mach. Volume 19 - Février 1976

[17] B. BHANU
Shape matching and image segmentation using stochastic labelling
SRI - USCIPI - Report 1030 - August 81

[18] R.O. DUDDA
Use of Hough transformation to detect lines and curves in picture
Commun Ass Computer Mach.- Vol. 15 - Janvier 72

[19] D. JUVIN - B. DUPEYRAT
ANIMA, a quasi real time system
Proceedings of IEEE, PRIP 82 - Las Vegas - June 1982

[20] U. RAMER
An iterative procedure for the polygonal approximation of plane curves
Computer Graphics and Image Processing 1 (3) Nov. 72

[21] J.T. TOU - R.C. GONZALES
Pattern recognition principles
Addison Wesley - 1974

[22] V.A. KOVALEVSKY
Image pattern recognition
Springer Verlag - 1980

[23] A. ROSENFELD - A.C. KAK
Digital Picture Processing
Academic Press - 1976

[24] R. HARANI - R. BORNE
Les bases de données images
C.N.A.M. Vol. C Paris - Décembre 1982

[25] D. JUVIN
Etude d'un transformateur de coordonnées pour MA23
Rapport Int DEIN/SIR/81073 - Juin 1980

[26] Etat de la robotique en France - Tomes I et II
Hermès - A.F.R.I. 2-86601-014-0 - 1982

/27/ M. HU
Visual pattern recognition by invariant moments
I.R.E. Transactions on information theory IT8 - 1962

/28/ J. WILF - R. CUNNINGHAM
Computing arbitrary moments from chain coded boundaries
Publication 7949 J.P.L. Pasadena - Nov. 1979

/29/ R. WONG - E. HALL
Scene matching with invariant moments
Computer Graphics and Image Processing - Volume 8 - 1978

/30/ Manuel de référence du langage ROL - Version 2
GIXI - Janvier 1983

/31/ B. DUPEYRAT
Contribution à la reconnaissance automatique de la parole
Analyse du signal temporel, reconnaissance de mots-isolés et identification de phénomènes
Thèse d'Etat, Paris VI, 1981

/32/ M. BAUDRY
Etude du signal vocal dans sa représentation amplitude-temps. Algorithmes de segmentation et de reconnaissance de la parole.
Thèse d'Etat, Paris VI, 1978

/33/ B. DUPEYRAT - D. JUVIN
ANIMA, un système d'analyse d'images
Journées de travail de l'AFCET à Arcueil, 2 et 3 Déc. 82

/34/ D. GENNERY - R. CUNNINGHAM
Computer vision
J.P.L. Publication 8192 - Nov. 1981

[35] R.C. BOLLES - R.A. CAIN
Recognizing and locating partially visible objects : The local-feature-focus method
S.R.I. Technical note 262 - Juin 1982

[36] R.C. BOLLES - J.H KREMERS - R.A. CAIN
A simple sensor to gather three-dimensional data
SRI technical note 249 - Juillet 1981

[37] H.J. LEVESQUE
A formal treatment of incomplete knowledge Data Bases
Fairchild Technical - Report 614 - Fev. 1982

[38] T.O. BINFORD - H.H. BAKER
Depth from image and intensity based stereo
Proceedings of 7th joint conference on Artificial Intelligence.
Vancouver - Aout 1981

[39] R.A. BROOKS
Model based 3D interpretations of 2D images
S.A.I.L. Publication

[40] T.O. BINFORD - D.G. LOWE
The interpretation of three dimensional structure from image curves
S.A.I.L. Publication - 1982

[41] F.M.A. VILNROTTER
Structural analysis of natural textures
University of Southern California USCIPI 1040 - Septembre 1981

[42] S.U. LEE
Design of SVD/SGK Convolution filters for image processing.
Univ of Southern California USCIPI 950 - Janvier 1980

/43/ J. BAJON - M. CATTOEN - I. RAKHODAI
Extracteur de caractéristiques d'images vidéo destiné à la robotique
Actes du 2ème Congrès AFCET-IRIA - Tome III - Sept. 1979

/44/ W. GEISLER
A vision system for shape and position recognition of industrial parts
Proc of the 2nd International Conference on Robot Vision
Stuttgart - Nov. 1982

Manuscrit reçu le 25 février 1983

ANNEXE A

UN SYSTÈME DE TRIANGULATION

Une méthode simple de mesure de relief z = f(x,y) est la triangulation optique par laser.

Cette méthode consiste à pointer à l'aide d'un capteur de lumière très directionnel (cellule photo-électrique + collimateur) le lieu où la mesure est désirée et, à l'aide d'un faisceau de lumière (laser) dont l'axe est différent de celui du capteur, chercher pour quelle incidence du faisceau on détecte la tache de diffusion sur l'objet.

La profondeur de l'objet est liée à l'angle d'incidence θ par une formule géométrique très simple.

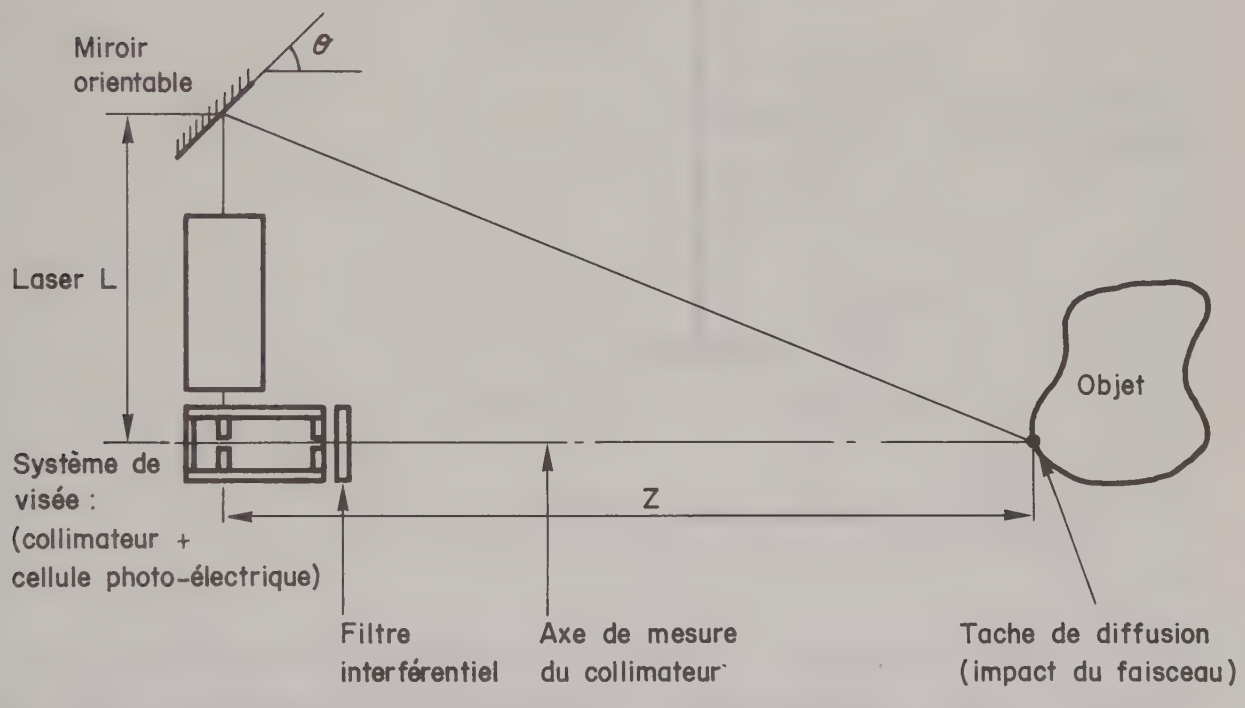


Figure A.1 _ Un système de triangulation laser Z = L tg (2 θ).

Nous avons expérimenté un tel système à l'aide d'une carte microprocesseur 8085.

Le miroir orientable est entraîné à vitesse constante (10 tours/sec). Un capteur potentiométrique indique, à tout moment, la position angulaire du miroir.

La cellule photo-électrique est reliée à un convertisseur Analogique-Digital est transmet à tout moment la valeur du signal mesuré.

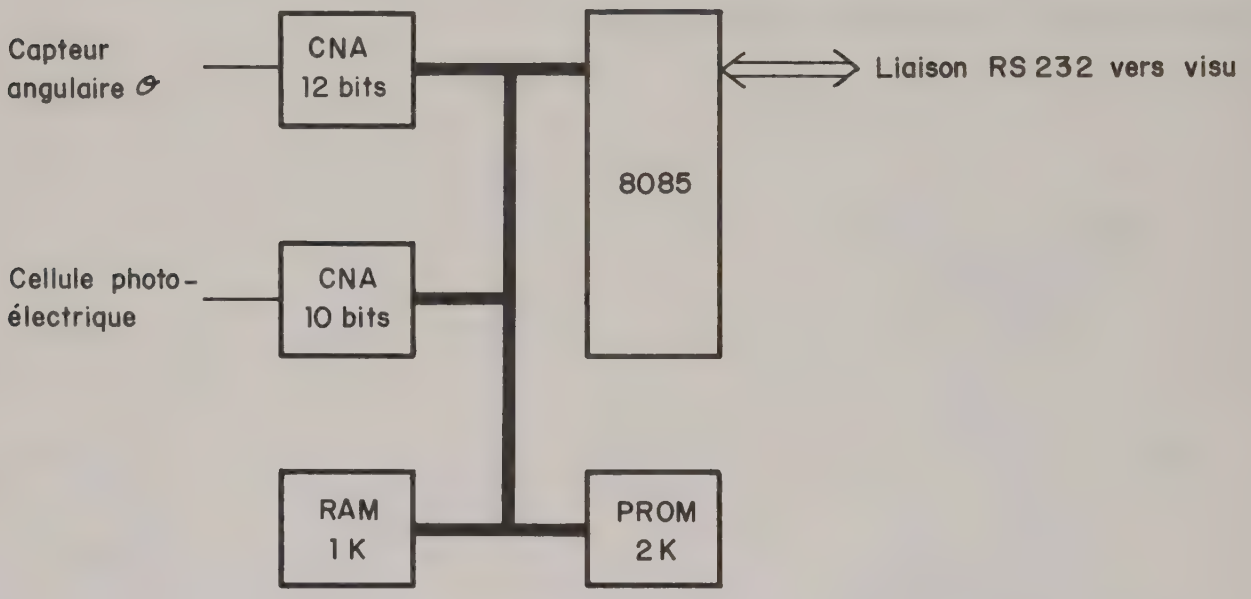


Figure A.2 _ Configuration matérielle.

Le programme de mesure consiste à déclencher simultanément la conversion sur les deux CNA et à attendre l'apparition d'un pic de tension sur la cellule correspondant au passage du spot dans le champ du collimateur. A ce moment, on sauvegarde la valeur de l'angle θ et on commence le calcul de Z. Après édition du résultat (sortie sur la visu), on retourne dans la phase d'attente du pic.

Ce petit système très simple à réaliser donne une estimation de la profondeur Z précise à environ 5‰ pour des distances comprises entre 5 cm et 4,50 m et présente l'avantage, par rapport aux systèmes ultra-soniques, d'être très directif (l'angle de mesure du collimateur est bien inférieur au degré).

Le système mesurait 10 points par seconde mais nous étions limités par la mécanique simplifiée due à une réalisation de l'ensemble en un temps très limité.

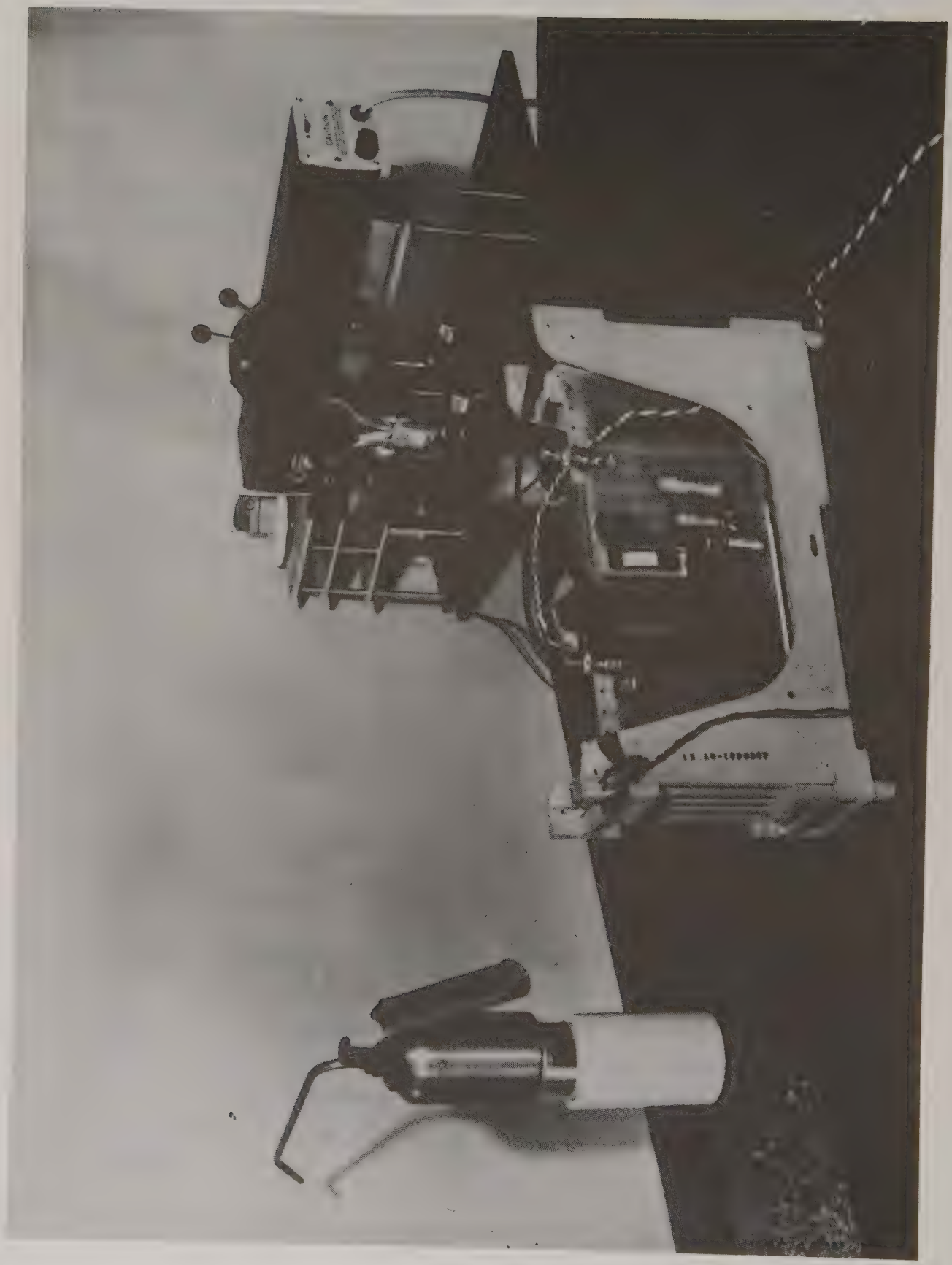


FIGURE A3 : Une photo du système

ANNEXE B

UTILISATION EN 3D DE L'ALGORITHME DE SUIVI DE CONTOUR

L'algorithme décrit en III.3. peut s'appliquer sur n'importe quelle fonction spatiale de forme :

$$a = f(x,y)$$

En particulier, dans le cas de l'image 3D :

$$z = f(x,y)$$

z exprimant la profondeur du point (x,y).

Cet algorithme détectera les variations brutales de relief. Ce qui correspond à la notion de ligne de contour dans une image 2D, se traduira ici par des lignes à forte transition de relief. Celles-ci sont très intéressantes pour de l'analyse de scène 3D car elles correspondent toujours à des lignes de forte pertinence :

- lignes du bord de l'objet (silhouette)
- certaines arêtes vives ou protubérances situées sur l'objet même.

(En fait,
il serait encore plus intéressant d'appliquer cet algorithme à la dérivée spatiale du relief :

$$z' = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}$$

car, dans ce système de représentation, on exprime l'orientation de la surface élémentaire en fonction de sa localisation, et les lignes de contours correspondraient aux lignes de brutales variations d'orientation (cas des arêtes)).

L'Institut National de Recherche Informatique et Automatique (INRIA) de Rocquencourt à réalisé un télémètre laser rapide et dispose d'une base d'images 3D selon le format de 128 x 128, z étant exprimé en réels (32 bits). Nous disposons de quelques images aimablement prêtées par Monsieur Faugeras afin de tester notre algorithme.

Notre système étant dépourvu de co-processeur flottant, nous avons dû convertir la représentation réelle en octets : la perte de précision résultante n'est pas trop pénalisante étant donnée la dynamique de l'image 3D originale.

Il nous a fallu modifier le module logiciel d'interface avec la mémoire d'image afin de pouvoir visualiser sur 4 bits l'image qui est traitée sur 8 bits.

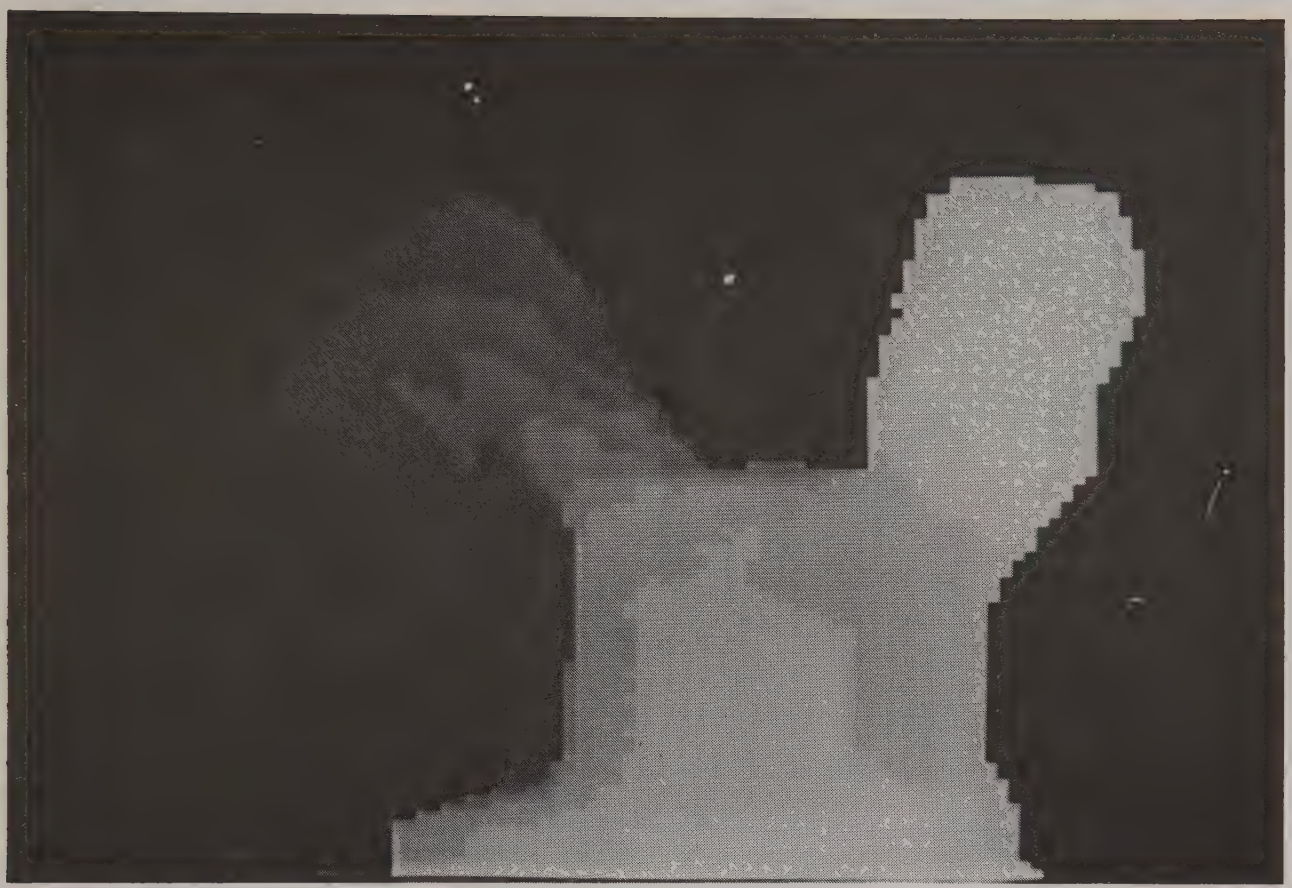
Nous représentons visuellement l'image 3D en faisant correspondre intensité et distance : c'est-à-dire qu'un point apparaîssant brillant sur l'image sera situé près du capteur et un point sombre sera loin de celui-ci. La figure B1 montre l'image ainsi représentée ainsi que les contours détectés sur l'image.

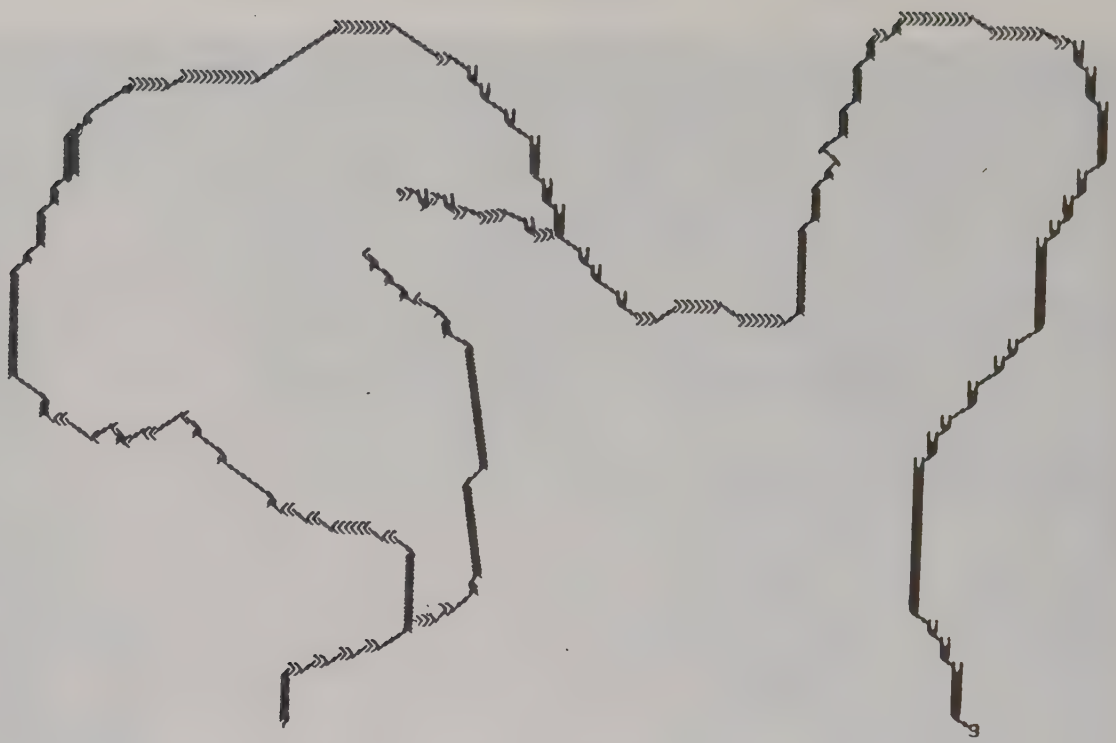
Nous avons appliqué notre opérateur de suivi dynamique à cette image 3D avec trois seuils différents :

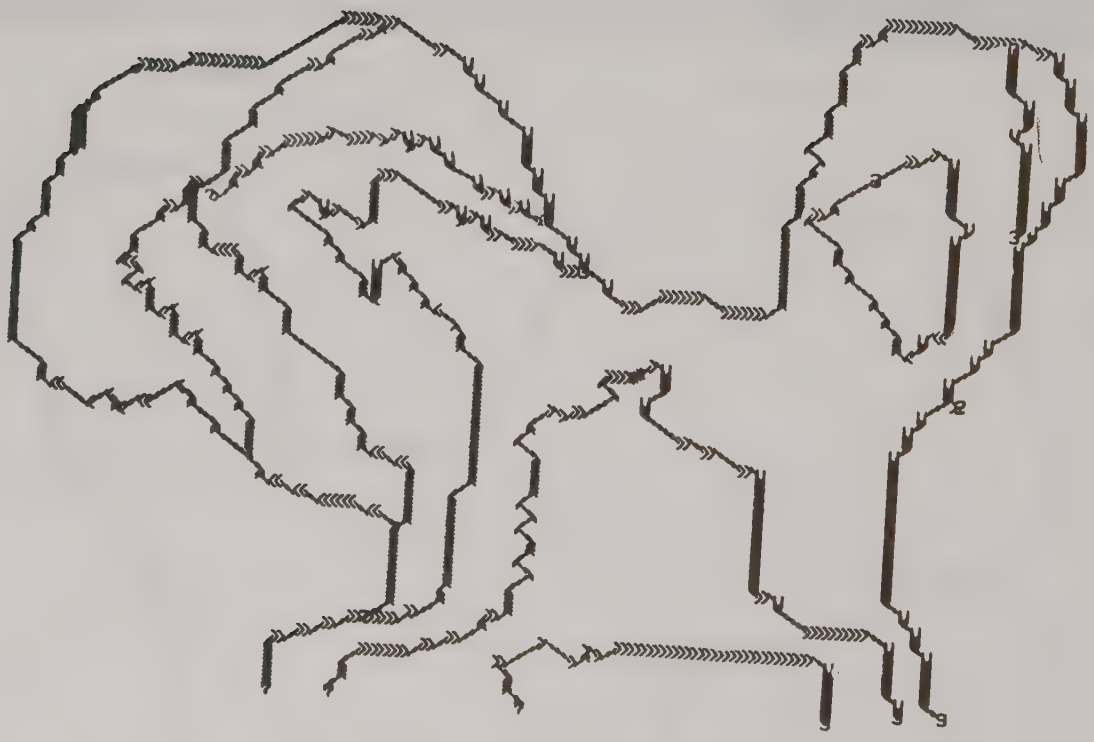
- un seuil élevé a donné la silhouette de l'objet (transitions brutales entre le fond et l'objet)

- un seuil moyen a indiqué 2 lignes de "cassures" correspondants à des changements brutaux de direction : (transitions importantes dans les lignes de niveaux)

- un seuil très bas indique les transitions progressives dans les lignes de niveaux.







ANNEXE C

PROPRIÉTÉ EUCLIDIENNE DE LA DISTANCE ENTRE SIGNATURES NORMALISÉES ET DE MÊME LONGUEUR

Nous avons défini (V.2.22) la distance entre deux signatures normalisées et de même longueur par la somme en valeur absolue des surfaces délimitées par les segments de leur codage développé.

Considérons deux signatures A et B normalisées et de même longueur et soit θ_p un angle tel que :

$$\theta_p \;/\; \forall \rho \in [\,0,\ \text{Périm}\,] \qquad \theta_A(\rho) > \theta_p \qquad \theta_B(\rho) > \theta_p$$

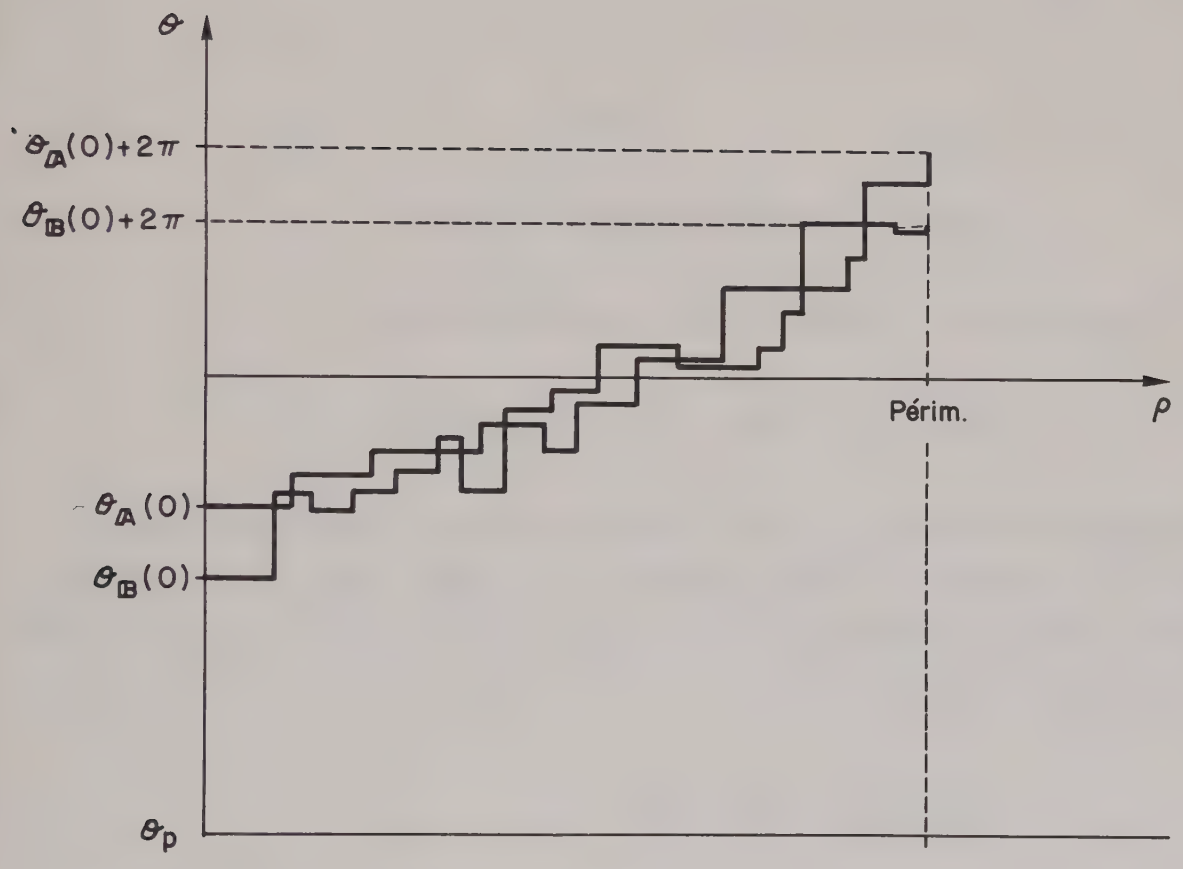


Figure C.1 _ Superposition de 2 signatures A et B normalisées et de même longueur.

Si l'on considère la forme $\mathcal{F}_A$ délimitée par les courbes d'équation :

$$\theta = \theta_p$$

$$\rho = 0$$

$$\theta = \theta_A(\rho)$$

$$\rho = \text{Périm}$$

et la forme $\mathcal{F}_B$ délimitée par les quatre courbes :

$$\theta = \theta_p$$

$$\rho = 0$$

$$\theta = \theta_B(\rho)$$

$$\rho = \text{Périm.}$$

Si l'on considère la distance d'aire définie par :

$$d'(\mathcal{F}_A, \mathcal{F}_B) = S_{\mathcal{F}_A} + S_{\mathcal{F}_B} - S_{\mathcal{F}_A \cap \mathcal{F}_B}$$

nous pouvons montrer que cette distance entre formes est équivalente à la distance entre signatures. En effet, ces deux formes ont trois côtés confondus, donc l'aire délimitée est nulle. L'aire entre ces deux formes est l'aire délimitée par les deux courbes $\theta_A(\rho)$ et $\theta_B(\rho)$, donc :

$$d'(\mathcal{F}_A, \mathcal{F}_B) = d(A, B)$$

Or, la distance d' vérifie les propriétés de distance Euclidienne (cf. V.2.22).

La distance entre-signatures d vérifie donc la propriété Euclidienne. c.q.f.d.

ANNEXE D

EXTRAITS DU PROGRAMME ANIMA

```
SERIES-III PL/M-86 V2.0 COMPILATION OF MODULE ANIMA
OBJECT MODULE PLACED IN :F1:ANIMAZ.OBJ
COMPILER INVOKED BY:  PLM86.86 :F1:ANIMAZ.SRC LARGE OPTIMIZE(3)


                /* ANIMA z AU 26/10/82 */
   1            ANIMA: DO;

                $NOLIST INCLUDE(:F1:PLMLIB.SRC)
                $NOLIST INCLUDE(:F1:MASLIB.SRC)
                $NOLIST INCLUDE(:F1:IMZLIB.SRC)
                $NOLIST INCLUDE(:F1:CALLIB.SRC)
                $NOLIST INCLUDE(:F1:COZLIB.SRC)
                $NOLIST INCLUDE(:F1:SEZLIB.SRC)
                $NOLIST INCLUDE(:F1:CRZLIB.SRC)
                $NOLIST INCLUDE(:F1:SYZLIB.SRC)
                /* Gestion dico */
 218    1           DCL LDICOMAX LIT'10';
 219    1           DCL LNOM LIT'16';DCL DEMILNOM LIT'8';
 221    1           DCL OBJ(10) STRUC (NOM(LNOM) BYTE,ISCAR(2) WORD,ISMX WORD,PERM WORD,
                    ADMXANG WORD,JMX BYTE,LSEG(20) WORD,AB(20) WORD,OD(20) INTG,
                    R(128) WORD,T(128) INTG,COINC(7) BYTE) PUBLIC;

                /* ANIMAZ */
 222    1           DCL COM$INCON LIT'OFFFFH';
 223    1           DCL OK LIT'0';
 224    1           DCL COMMANDE$STOP LIT'14';
 225    1           DCL (ADSV,ADSYS) WORD PUBLIC;
 226    1           DCL (PMESS,PTAM) PTR;
 227    1           DCL MESS BASED PMESS(1) WORD;
 228    1           DCL MOUT(256) WORD;

 229    1           DCL (ATT,BX,BYY,GRO,LGEFF,LMESS,IDOCC,J,K,LONG,LRO,ROO,ROD,ROEC,ROMAX,SCOR,SDIV,SEUILTEST) WORD;
 230    1           DCL (BPIX,CERCLE,CPIX,CPTR,ER,ERR,EXIST,FAC,FACT,I,ID,II,IJ,JK,PAS,PLEIN,TAILLE,TPIX) BYTE;
 231    1           DCL (VIERGE) BYTE;
 232    1           DCL (ECTHMX,GTHETA,THEC,THETAMAX) INTEGER;
 233    1           DCL PRIS(4) WORD;

 234    1           DCL LONG$COMMANDE LIT'18';
 235    1           DCL COMMANDE(LONGCOMMANDE) BYTE DATA(11,14,1,2,6,5,15,10,13,7,4,9,19,16,3,12,8,17);
 236    1           DCL ML(2) BYTE DATA (0CCH,0C3H);
 237    1           DCL AMONIT WORD DATA (.ML);

                /*************************************************************************/
                /*                                                                       */
                /*                        Gestion du dictionnaire                        */
                /*                                                                       */
                /*************************************************************************/

 238    1           GESTDICO:   PROC(PNOM,PPLEIN,PEXIST) BYTE;
                    /* A partir d'une chaine de 16 caracteres,recherche de son occurence
                    dans le dico */

 239    2               DCL (PEXIST,PNOM,PPLEIN) PTR;
 240    2               DCL NOM BASED PNOM(1) BYTE;
 241    2               DCL PLEIN BASED PPLEIN BYTE;
```

```
242  2            DCL EXIST BASED PEXIST BYTE;
243  2            DCL I BYTE;
244  2            DO I=1 TO IMAX;
245  3                IF CMPB(@OBJ(I-1).NOM(0),PNOM,LNOM)=0FFFFH THEN DO;
247  4                    EXIST=VRAI;
248  4                    PLEIN=FAUX;
249  4                    RETURN I-1;
250  4                    END;
251  3                END;
252  2            EXIST=FAUX;
253  2            IF IMAX<LDICOMAX THEN DO;
255  3                IMAX=IMAX+1;
256  3                PLEIN=FAUX;
257  3                END;
258  2            ELSE PLEIN=VRAI;
259  2            RETURN IMAX-1;
260  2            END GESTDICO;

261  1        ENREGIST:   PROC(PNOM) BYTE;
262  2            DCL PNOM PTR;
263  2            DCL NOM BASED PNOM(1) BYTE;
264  2            DCL (I,J,EXIST,PLEIN) BYTE;
265  2            IF NOM(0)=' ' THEN RETURN 3;
267  2            I=GESTDICO(PNOM,@PLEIN,@EXIST);
268  2            IF PLEIN THEN RETURN 2;
270  2            CALL MOVB(PNOM,@OBJ(I).NOM(0),LNOM);
271  2            CALL MOVW(@ISP(0),@OBJ(I).ISCAR(0),2);
272  2            OBJ(I).ISMX=ISMAX;
273  2            OBJ(I).PERM=PERIM;
274  2            OBJ(I).ADMXANG=ABGTHETA;
275  2            OBJ(I).JMX=JMAX;
276  2            DO J=0 TO JMAX;
277  3                OBJ(I).LSEG(J)=LONGSEG(J);
278  3                OBJ(I).AB(J)=ABSC(J);
279  3                OBJ(I).OD(J)=ORD(J);
280  3                END;
281  2            DO K=0 TO ISMAX;
282  3                OBJ(I).R(K)=RO(K);
283  3                OBJ(I).T(K)=THETA(K);
284  3                END;
285  2            IF EXIST THEN RETURN 6;ELSE RETURN 0;
288  2            END ENREGIST;

        /*************************************************************************/
        /*                                                                       */
        /*                        SOUS PROGRAMME DELETE                          */
        /*                                                                       */
        /*************************************************************************/

289  1        DELETE: PROC(PNOM) WORD;
290  2            DCL PNOM PTR;
291  2            DCL NOM BASED PNOM(1) BYTE;
292  2            DCL (I,PLEIN,EXIST) BYTE;
293  2            I=GESTDICO(PNOM,@PLEIN,@EXIST);
294  2            IF NOT EXIST THEN RETURN 3;
296  2            IMAX=IMAX-1;
297  2            IF I<>IMAX THEN DO;
```

```
299   3                 CALL MOVB(@OBJ(IMAX).NOM(0),@OBJ(I).NOM(0),LNOM);
300   3                 CALL MOVW(@OBJ(IMAX).ISCAR(0),@OBJ(I).ISCAR(0),2);
301   3                 OBJ(I).ISMX=OBJ(IMAX).ISMX;
302   3                 OBJ(I).PERM=OBJ(IMAX).PERM;
303   3                 OBJ(I).ADMXANG=OBJ(IMAX).ADMXANG;
304   3                 OBJ(I).JMX=OBJ(IMAX).JMX;
305   3                 DO J=0 TO JMAX;
306   4                     OBJ(I).LSEG(J)=OBJ(IMAX).LSEG(J);
307   4                     OBJ(I).AB(J)=OBJ(IMAX).AB(J);
308   4                     OBJ(I).OD(J)=OBJ(IMAX).OD(J);
309   4                     END;
310   3                 DO K=0 TO ISMAX;
311   4                     OBJ(I).R(K)=OBJ(IMAX).R(K);
312   4                     OBJ(I).T(K)=OBJ(IMAX).T(K);
313   4                     END;
314   3                 END;
315   2             RETURN 0;
316   2             END DELETE;


              /*****************************************************************************/
              /*                                                                           */
              /*                                  PROGRAMME PRINCIPAL                      */
              /*                                                                           */
              /*****************************************************************************/

                  /* INITIALISATIONS */
317   1           DISABLE;
318   1           ADSV=RLOOKUP(@(6,'VISION'),@CPTR);
319   2           IF CPTR()0 THEN DO;ER=10H;GO TO ETRANG;END;
324   1           ADSYS=RLOOKUP(@(7,'SYSTEME'),@CPTR);
325   2           IF CPTR()0 THEN DO;ER=20H;GO TO ETRANG;END;
330   1       EN: PMESS=RRECEIVE(ADSV,0FFFFH,@CPTR);
331   2           IF CPTR()0 THEN DO;ER=30H;GO TO ETRANG;END;
                  /* Decodage */

336   1       RELEN:  MOUT(1)=ADSV;MOUT(2)=ADSYS;
338   1           ATT=0;
339   1           DO CASE (FINDB(@COMMANDE(0),MESS(3),LONGCOMMANDE)+1);
340   2               DO; /* Commande inconnue */
341   3                   LMESS=8;MOUT(3)=COM$INCON;
343   3                   END;
344   2               DO; /* Initialisation */
345   3                   CALL ORG;
346   3                   BPIX=16;
347   3                   CPIX=1;
348   3                   TPIX=16;
349   3                   VIERGE=VRAI;
350   3                   NIVMINDEC=11;J=0;
352   3                   ROMAX=160;
353   3                   THETAMAX=1600;
354   3                   PAS=16;
355   3                   FACT=15;
356   3                   ECTHMX=80;
357   3                   XMX=6;
358   3                   IMAX=0;
359   3                   SEUILMIN=30;SEUIL=23;
```

```
361  3                  LINMIN=32;
362  3                  LIGNE=200;
363  3                  ATT=0;
           /*           IF LIGNE)LINMIN THEN IDOCC=SHL(LIGNE-LINMIN,4);ELSE IDOCC=0;
                        IF(LIGNE+LINMIN)(256 THEN LGEFF=SHL(LINMIN,5)+15;ELSE LGEFF=4095-IDOCC;
           */
364  3                  IDOCC=1008;
365  3                  LGEFF=1039;
366  3                  LMESS=10;MOUT(3)=OK;MOUT(4)=OFFFFH;
369  3                  SDIV=150;
370  3                  FAC=30;
371  3                  END;
372  2                DO; /* Retour moniteur (prov.) */
373  3                  CALL AMONIT;
374  3                  LMESS=8;MOUT(3)=OK;
376  3                  END;
377  2                DO; /* Acquisition */
378  3                  VIERGE=VRAI;
379  3                  CALL ACQUIERS(DOUBLE(0),4095);
380  3                  LMESS=10;MOUT(3)=OK;MOUT(4)=0;
383  3                  ATT=20;
384  3                  END;
385  2                DO; /* Acquisition rapide */
386  3                  VIERGE=VRAI;
387  3                  CALL ACQUIERS(IDOCC,LGEFF);
388  3                  LMESS=10;MOUT(3)=OK;MOUT(4)=0;
391  3                  ATT=20;
392  3                  END;
393  2                DO; /* Clear */
394  3                  VIERGE=VRAI;
395  3                  IF MESS(4)()0 THEN CPIX=MESS(4);
397  3                  CALL EFFACE(CPIX-1);
398  3                  LMESS=8;MOUT(3)=OK;
400  3                  ATT=20;
401  3                  END;
402  2                DO; /* Cherche */
403  3                  IF MESS(4)()0 THEN DO;
405  4                      SEUILMIN=MESS(4);SEUIL=SEUILMIN-SHR(SEUILMIN,2);
407  4                      END;
408  3                  IF MESS(5)()0 THEN DO;
410  4                      LIGNE=LOW(MESS(5));
411  4                      IF LIGNE)LINMIN THEN IDOCC=SHL(LIGNE-LINMIN,4);ELSE IDOCC=0;
414  4                      IF(LIGNE+LINMIN)(256 THEN LGEFF=SHL(LINMIN,5)+15;ELSE LGEFF=4095-IDOCC;
417  4                      END;
418  3                  CALL CONTOURZ; /* CONTOURZ se met en attente d'une commande */
419  3                  IF CONTOUR$TROUVE THEN DO;
421  4                      IF ARRET THEN DO;
423  5                          IF MESS(3)=COMMANDE$STOP THEN DO;
425  6                              LMESS=8;MOUT(3)=OK;
427  6                              MOUT(1)=ADSV;MOUT(2)=ADSYS;
429  6                              END;
430  5                          ELSE DO;
431  6                              MOUT(3)=0;GO TO RELEN;
433  6                              END;
434  5                          END;
435  4                      ELSE DO;
436  5                          LMESS=10;MOUT(4)=OFFFFH;
```

```
438  5                    IF FERME THEN MOUT(3)=0;ELSE MOUT(3)=1;
441  5                    END;
442  4                END;
443  3            ELSE DO; /* Contour non present */
444  4                LMESS=10;MOUT(4)=1;MOUT(3)=1;
447  4                END;
448  3            END;
449  2        DO; /* Trace */
450  3            LMESS=8;
451  3            IF MESS(4)<>0 THEN TPIX=MESS(4);
453  3            MOUT(3)=TRACE(TPIX-1);
454  3            END;
455  2        DO; /* Extrapolation */
456  3            LMESS=8;
457  3            MOUT(3)=EXTRAI(DOUBLE(1));
458  3            CALL EXAF;
459  3            END;
460  2        DO; /* Selection des points de prise */
461  3            LMESS=8;
462  3            MOUT(3)=PRISE;
463  3            END;
464  2        DO; /* Codage relatif */
465  3            LMESS=8;
466  3            IF FERME THEN DO;
468  4                MOUT(3)=OK;
469  4                CALL CORDON;
470  4                END;
471  3            ELSE MOUT(3)=1;
472  3            END;
473  2        DO; /* Calcul des caracteristiques */
474  3            IF FERME THEN DO;
476  4                IF MESS(4)<>0 THEN ROMAX=MESS(4);
478  4                IF MESS(5)<>0 THEN THETAMAX=SIGNED(MESS(5));
480  4                ISMAX=CALMAX(ROMAX,THETAMAX,FERME,ISD,@GRO,@GTHETA)-1;
481  4                SENSTRIGO=(THETA(ISMAX)-THETA(ISD+1))>0);
482  4                PERIM=RO(ISMAX);
483  4                ECARCIR=CIRCUL(ISD,PERIM,PAS);
484  4                IF PERIM<3600 THEN SEUILTEST=125-PERIM/60;
486  4                ELSE SEUILTEST=65;
487  4                IF ECARCIR<SEUILTEST THEN DO;
489  5                    CERCLE=VRAI;
490  5                    LMESS=12;
491  5                    MOUT(3)=7;
492  5                    MOUT(4)=PERIM;
493  5                    MOUT(5)=SHL(PERIM,2)/251;
494  5                    END;
495  4                ELSE DO;
496  5                    CERCLE=FAUX;
497  5                    LMESS=8;
498  5                    MOUT(3)=OK;
499  5                    CALL GROSSEG(DOUBLE(0),ISMAX,VRAI,PERIM,@GRO,@JMAX);
500  5                END;
501  4            END;
502  4            ELSE DO;LMESS=8;MOUT(3)=1;END;
506  3            END;
507  2        DO; /* Enregistrement */
508  3            IJ=4;
```

```
509   4                 DO WHILE(MESS(IJ))255) AND (IJ(12);IJ=IJ+1;END;
512   4                 IF(MESS(IJ)()0) AND (MESS(IJ)(255) THEN DO;MESS(IJ)=MESS(IJ) OR 2000H;IJ=IJ+1;END;
517   3                 DO JK=IJ TO SHR(LNOM,1)+4;
518   4                     MESS(JK)=2020H;
519   4                     END;
520   3                 LMESS=8;
521   3                 MOUT(3)=ENREGIST(@MESS(4));
522   3                 END;
523   2             DO; /* Delete */
524   3                 IJ=4;
525   4                 DO WHILE(MESS(IJ))255) AND (IJ(12);IJ=IJ+1;END;
528   4                 IF(MESS(IJ)()0) AND (MESS(IJ)(255) THEN DO;MESS(IJ)=MESS(IJ) OR 2000H;IJ=IJ+1;END;
533   3                 DO JK=IJ TO SHR(LNOM,1)+4;
534   4                     MESS(JK)=2020H;
535   4                     END;
536   3                 LMESS=8;
537   3                 MOUT(3)=DELETE(@MESS(4));
538   3                 END;
539   2             DO; /* Tri sur caracteristiques */
540   3                 IF IMAX()0 THEN DO;
542   4                     IF MESS(4)()0 THEN FACT=MESS(4);
544   4                     IF MESS(5)()0 THEN ECTHMX=SIGNED(MESS(5));
546   4                     IF MESS(6)()0 THEN XMX=MESS(6)-1;
548   4                     IRECON=RECON(FACT,ECTHMX,XMX,PERIM,@COINCIDMX,@XRECON);
549   4                     JK=4;
550   4                     CALL MOVB(@OBJ(IRECON).NOM(0),@MOUT(4),LNOM);
551   4                     JK=DEMILNOM+JK;
552   4                     MOUT(JK)=COINCIDMX;
553   4                     MOUT(JK+1)=XRECON;
554   4                     MOUT(JK+2)=XMX;
555   4                     JK=JK+3;
556   4                     DO IJ=0 TO IMAX-1;
557   5                         CALL MOVB(@OBJ(IJ).NOM(0),@MOUT(JK),LNOM);
558   5                         JK=JK+DEMILNOM;
559   5                         DO II=0 TO XMX;
560   6                             MOUT(JK)=OBJ(IJ).COINC(II);
561   6                             JK=JK+1;
562   6                             END;
563   5                         END;
564   4                     MOUT(3)=0;
565   4                     LMESS=SHL(JK,1);
566   4                     END;
567   4                 ELSE DO;LMESS=6;MOUT(3)=1;END;
571   3                 END;
572   2             DO; /* Baratin */
573   3                 IF VIERGE THEN DO;
575   4                     VIERGE=FAUX;
576   4                     BX=5;BYY=20;TAILLE=2;BPIX=16;
580   4                     END;
581   3                 IF MESS(22)()0 THEN BX=MESS(22);
583   3                 IF MESS(23)()0 THEN BYY=MESS(23);
585   3                 IF MESS(24)()0 THEN TAILLE=MESS(24);
587   3                 IF MESS(25)()0 THEN BPIX=MESS(25);
589   3                 CALL BARATIN(BX,@BYY,BPIX-1,TAILLE-1,@MESS(4));
590   3                 BYY=BYY+SHL(TAILLE+1,3)+4;
591   3                 LMESS=8;
592   3                 MOUT(3)=OK;
```

```
593   3                 END;
594   2               DO; /* Learn */
595   3                 IF EXTRAI(DOUBLE(1))=1 THEN DO;
597   4                     LMESS=8;MOUT(3)=1;
599   4                     END;
600   3                 ELSE DO;
601   4                     CALL CORDON;
602   4                     ISMAX=CALMAX(ROMAX,THETAMAX,FERME,ISD,@GRD,@GTHETA)-1;
603   4                     SENSTRIGO=(THETA(ISMAX)-THETA(ISD+1))>0);
604   4                     PERIM=RO(ISMAX);
605   4                     ECARCIR=CIRCUL(ISD,PERIM,PAS);
606   4                     IF PERIM<3600 THEN SEUILTEST=125-PERIM/60;
608   4                     ELSE SEUILTEST=65;
609   4                     IF ECARCIR<SEUILTEST THEN DO;
611   5                         CERCLE=VRAI;
612   5                         LMESS=12;
613   5                         MOUT(3)=7;
614   5                         MOUT(4)=PERIM;
615   5                         MOUT(5)=SHL(PERIM,2)/251;
616   5                         END;
617   4                     ELSE DO;
618   5                         CERCLE=FAUX;
619   5                         CALL GROSSEG(DOUBLE(0),ISMAX,VRAI,PERIM,@GRD,@JMAX);
620   5                         IJ=4;
621   6                         DO WHILE(MESS(IJ))255) AND (IJ<12);IJ=IJ+1;END;
624   6                         IF(MESS(IJ)<>0) AND (MESS(IJ)<255) THEN DO;MESS(IJ)=MESS(IJ) OR 2000H;IJ=IJ+1;END;
629   5                         DO JK=IJ TO SHR(LNOM,1)+4;
630   6                             MESS(JK)=2020H;
631   6                             END;
632   5                         LMESS=8;
633   5                         MOUT(3)=ENREGIST(@MESS(4));
634   5                         IF NOT APPREN THEN MOUT(3)=5;
636   5                         END;
637   4                     END;
638   3                 END;
639   2               DO; /* Correlation */
640   4                 IF CERCLE THEN DO;LMESS=8;MOUT(3)=7;END;
645   3                 ELSE DO;
646   4                     IJ=4;
647   4                     IF MESS(IJ)='?' THEN DO; /* Correlation auto */
649   5                         IF MESS(12)<>0 THEN SDIV=MESS(12);
651   5                         IF MESS(13)<>0 THEN FAC=MESS(13);
653   5                         IF(MOUT(3):=INSPEC(SDIV,FAC,@PRIS(0),@THEC,@SCOR,@ID))=0
654   5                         THEN DO;
655   6                             LMESS=20;
656   6                             CALL MOVW(@PRIS(0),@MOUT(4),4);
657   6                             MOUT(8)=UNSIGN(THEC);
658   6                             MOUT(9)=SCOR;
659   6                             END;
660   5                         ELSE LMESS=8;
661   5                         END;
662   4                     ELSE DO; /* Correlation manuelle */
663   6                         DO WHILE(MESS(IJ))255) AND (IJ<12);IJ=IJ+1;END;
666   5                         IF (MESS(IJ)<>0) AND (MESS(IJ)<255) THEN DO;
668   6                             MESS(IJ)=MESS(IJ) OR 2000H;
669   6                             IJ=IJ+1;
670   6                             END;
```

```
671   5                    DO JK=IJ TO SHR(LNOM,1)+4;
672   6                        MESS(JK)=2020H;
673   6                        END;
674   5                    I=GESTDICO(@MESS(4),@PLEIN,@EXIST);
675   5                    IF EXIST THEN DO;
677   6                        MOUT(3)=0;LMESS=20;
679   6                        ROO=MESS(12);ROD=MESS(13);
681   6                        IF MESS(14)<>0 THEN LONG=MESS(14);ELSE LONG=OBJ(I).PERM;
684   6                        IF MESS(15)<>0 THEN SDIV=MESS(15);
686   6                        IF MESS(16)<>0 THEN LRO=MESS(16);ELSE LRO=OBJ(I).PERM;
689   6                        SCOR=CORREL2(@OBJ(I).ISMX,@OBJ(I).R(0),
                               @OBJ(I).T(0),ROD,ROO,RO(ISMAX),LONG,LRO,SDIV,@ROEC,@THEC);
690   6                        CALL OUESTCE(ROEC,OBJ(I).R(OBJ(I).ISCAR(0)),
                               OBJ(I).R(OBJ(I).ISCAR(1)),OBJ(I).PERM,@PRIS(0));
691   6                        LMESS=20;MOUT(3)=0;
693   6                        CALL MOVW(@PRIS(0),@MOUT(4),4);
694   6                        MOUT(8)=UNSIGN(THEC);
695   6                        MOUT(9)=SCOR;
696   6                        END;
697   5                    ELSE DO;
698   6                        LMESS=8;
699   6                        MOUT(3)=3;
700   6                        END;
701   5                    END;
702   4                END;
703   3            END;

704   2            DO; /* Reconnaissance */
705   3                IF MESS(4)<>0 THEN SDIV=MESS(4);
707   3                IF (EXTRAI(DOUBLE(1))=1) OR (IMAX=0) THEN DO;
709   4                    LMESS=8;MOUT(3)=1;
711   4                    END;
712   3                ELSE DO;
713   4                    CALL CORDON;
714   4                    ISMAX=CALMAX(ROMAX,THETAMAX,FERME,ISD,@GRO,@GTHETA)-1;
715   4                    SENSTRIGO=(THETA(ISMAX)-THETA(ISD+1))>0;
716   4                    PERIM=RO(ISMAX);
717   4                    ECARCIR=CIRCUL(ISD,PERIM,PAS);
718   4                    IF PERIM<3600 THEN SEUILTEST=125-PERIM/60;
720   4                    ELSE SEUILTEST=65;
721   4                    IF ECARCIR<SEUILTEST THEN DO;
723   5                        CERCLE=VRAI;
724   5                        LMESS=38;
725   5                        MOUT(3)=7;
726   5                        JK=4;
727   5                        CALL MOVB(@('***  Cercle  ***'),@MOUT(4),LNOM);
728   5                        JK=JK+DEMILNOM;
729   5                        CALL PRICERCL(@MOUT(JK));
730   5                        JK=JK+4;
731   5                        MOUT(JK)=PERIM;
732   5                        MOUT(JK+1)=ECARCIR;
733   5                        MOUT(JK+2)=14;
734   5                        END;
735   4                    ELSE DO;
736   5                        CERCLE=FAUX;
737   5                        CALL GROSSEG(DOUBLE(0),ISMAX,VRAI,PERIM,@GRO,@JMAX);
738   5                        IRECON=RECON(FACT,ECTHMX,XMX,PERIM,@COINCIDMX,@XRECON);
```

```
 739   5                        IF(MOUT(3):=INSPEC(SDIV,FAC,@PRIS(0),@THEC,@SCOR,@ID))=0
 740   5                        THEN DO;
 741   6                            LMESS=38;
 742   6                            JK=4;
 743   6                            CALL MOVB(@OBJ(ID).NOM(0),@MOUT(JK),LNOM);
 744   6                            JK=JK+DEMILNOM;
 745   6                            CALL MOVW(@PRIS(0),@MOUT(JK),4);
 746   6                            JK=JK+4;
 747   6                            MOUT(JK)=UNSIGN(THEC);
 748   6                            MOUT(JK+1)=SCOR;
 749   6                            MOUT(JK+2)=COINCIDMX;
 750   6                            END;
 751   5                        ELSE LMESS=8;
 752   5                        END;
 753   4                    END;
 754   3                END;

 755   2            END; /* FIN DU DO CASE */
 756   1          CALL RFREESEGMENT(PMESS,@CPTR);
 757   2          IF CPTR()0 THEN DO;ER=40H;GO TO ETRANG;END;
 762   1          PTAM=RGETSEGMENT(LMESS,@CPTR);
 763   1          CALL MOVW(@MOUT(0),PTAM,SHR(LMESS,1));
 764   1          CALL RSEND(ADSV,ADSYS,PTAM,@CPTR);
 765   1          IF ATT()0 THEN CALL DELAI(ATT);
 767   1          GO TO EN;
 768   1      ETRANG: PTAM=RGETSEGMENT(DOUBLE(8),@CPTR);
 769   1          MOUT(3)=ERR;
 770   1          CALL MOVW(@MOUT(0),PTAM,4);
 771   1          CALL RSEND(ADSV,ADSYS,PTAM,@CPTR);
 772   1          CALL AMONIT;

 773   1          END ANIMA;
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 1346H   4934D
     CONSTANT AREA SIZE = 0000H      0D
     VARIABLE AREA SIZE = 1C5FH   7263D
     MAXIMUM STACK SIZE = 0026H     38D
     592 LINES READ
     0 PROGRAM WARNINGS
     0 PROGRAM ERRORS

END OF PL/M-86 COMPILATION
```

DIDIER JUVIN

CONTRIBUTION A LA RECONNAISSANCE AUTOMATIQUE DES IMAGES APPLIQUEE A LA ROBOTIQUE

Thèse de Docteur-Ingénieur soutenue le 31 Janvier 1983
à l'Université PARIS-SUD Centre d'ORSAY.

RESUME :

Ce mémoire présente une méthode d'analyse et d'interprétation d'images d'objets situés dans une scène simple constituant l'environnement d'un robot.

La première partie traite de l'extraction du contour des objets présents dans l'image et développe une technique originale de suivi de contours basée sur la notion d'arborescence de lignes combinée à une "fonction de coût" quantifiant la qualité du contour détecté. Nous présentons une heuristique de parcours à coût raisonnable de l'arborescence en un temps minimal, ce qui équivaut à suivre les lignes de contour probables dans l'image. Une technique de segmentation du contour invariante en translation et en rotation est ensuite exposée.

La deuxième partie présente une méthode de reconnaissance basée sur l'utilisation du codage invariant précédent: cet algorithme effectue un tri préalable sur des caractéristiques grossières extraites de la segmentation, suivi d'une comparaison entre les formes de probables identités par application d'une distance définie sur ce codage invariant.

La dernière partie décrit l'aboutissement de cette étude qui fait l'objet d'une application industrielle et va constituer le système de vision d'une gamme de robots. Ce système est implanté sur un micro-processeur 16 bits et fonctionne en temps réel.

MOTS CLES :

IMAGES (analyse d')
CONTOUR (extraction de)
RECONNAISANCE DE FORMES
ROBOTIQUE
CODAGE (de contours,de formes)